ZIPSPLAT: OBSERVATION-GROUNDED SCENE TOKENS FOR FEED-FORWARD GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

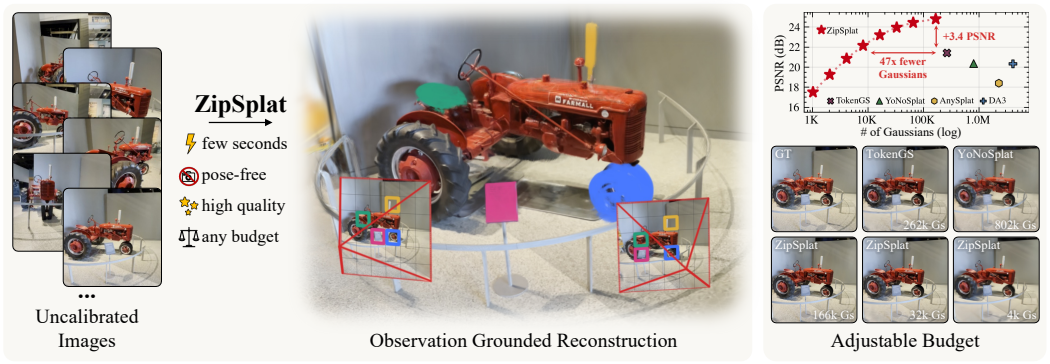


Figure 1: ZipSplat reconstructs scenes from unposed images in seconds. Each scene token is derived from a source image patch and decoded into freely placed Gaussians that are grounded in their observation (colored groups ↔ patches). Redundant tokens across views are merged into a compact set whose size is adjustable at inference without retraining. *Right:* the same DL3DV scene from 16 views at 166K, 32K, and 4K Gaussians. ZipSplat matches the strongest baseline with 47× fewer Gaussians and surpasses it by 3.4 dB at full budget.

ABSTRACT

Feed-forward 3D Gaussian Splatting reconstructs scenes from images in seconds. Pixel-aligned methods keep every Gaussian grounded in an observation, but produce redundant, potentially misaligned Gaussians across views. Token-based methods instead aggregate all views into a fixed set of learned tokens but lack explicit observation support and use a fixed reconstruction budget. We introduce ZipSplat, a method that predicts a compact, observation-grounded scene representation whose size is set at inference. It derives each scene token from an image patch rather than a learned query, and decodes it into freely placed 3D Gaussians. Tokens across views are merged into a compact set whose size is adjustable at inference. However, without geometric supervision, decoded Gaussians can drift from their source patch, and merging can mix tokens from different surfaces. We therefore introduce two geometric losses to ground and shape this representation. A cone loss ties each token's Gaussians to its source patch, and GeoCE aligns token similarity with 3D proximity so that tokens observing the same surface merge together. Building on these observation–geometry associations, we further introduce TokenBA, a token-level bundle adjustment that refines scene tokens and camera poses in a few seconds. ZipSplat requires neither camera poses nor intrinsics, yet outperforms pixel-aligned and token-based baselines on DL3DV and RealEstate10K, even those given ground-truth cameras. It further generalizes zero-shot to Mip-NeRF360 and ScanNet++. Code and trained models will be made public.

1 INTRODUCTION

Novel view synthesis has shifted from per-scene optimization of radiance fields (Mildenhall et al., 2020) and 3D Gaussians (Kerbl et al., 2023) to feed-forward models that predict a Gaussian scene from a collection of images in a few seconds (Charatan et al., 2024; Chen et al., 2024; Xu et al., 2025). As these methods mature, performance increasingly depends on the scene representation itself: how image observations become Gaussians, how many are predicted, and where they are placed.

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.

Most existing feed-forward methods predict one Gaussian per input pixel (Xu et al., 2025; Ye et al., 2025a; 2026; Jiang et al., 2025), directly grounding each prediction in an image observation. However, repeated observations of the same surface produce separate, view-specific Gaussians rather than a shared prediction, creating substantial cross-view redundancy. Errors in the depth and camera estimates can further misalign these predictions and produce visible artifacts.

Recent methods based on fixed latent queries instead aggregate multi-view observations into shared tokens before decoding freely placed Gaussians (An et al., 2026; Ren et al., 2026; Itkin et al., 2026). This aggregation can reduce redundancy and cross-view misalignment, but the learned queries have no explicit link to the observations they represent. Unlike pixel-aligned Gaussians, whose source view is known by construction, these tokens must learn their observation support implicitly, which can produce blurry reconstructions. Their reconstruction budget is also fixed by design: a small object capture and a large scene receive the same number of tokens and Gaussians.

These limitations motivate a representation that grounds each prediction in its source observation without fixing the reconstruction budget in the architecture. We introduce **ZipSplat**, which predicts a compact, observation-grounded scene representation from unposed images. Its reconstruction budget is adjustable at inference without retraining, letting it match the scene rather than the architecture.

Rather than using learned queries, we derive each scene token directly from its source image patch, making the token-to-observation link explicit by construction. Each token is decoded into freely placed 3D Gaussians, but without geometric constraints the decoded Gaussians can drift from the patch that produced them. A new cone loss addresses this by constraining each token's Gaussians to reproject near their source patch, grounding them in their observation.

As views overlap, many patches observe the same surface and their tokens can be merged into a more compact set. However, merging in feature space does not guarantee that the resulting groups correspond to the same surface. A new geometric contrastive loss, GeoCE, aligns token similarity with 3D proximity so that clusters form by surface rather than by viewpoint.

The representation is now compact and grounded, but it remains the output of a single forward pass, with residual camera and geometry error. The grounding that ties each token's Gaussians to its source patch also gives every token a persistent 2D–3D correspondence that can be used for geometric refinement. We exploit this with TokenBA, a token-level bundle adjustment that refines cameras and scene tokens through the frozen decoder, improving reconstruction in seconds.

ZipSplat establishes a new state of the art on standard benchmarks using only unposed, uncalibrated images at inference. It consistently outperforms both pixel-aligned and token-based feed-forward methods while using substantially fewer Gaussians. It generalizes zero-shot to diverse indoor and outdoor scenes, and continues to improve as additional views are provided. Ablations validate the representation and its components, and further experiments characterize the adjustable budget and cross-dataset generalization.

In summary, we make the following contributions:

- We introduce ZipSplat, a feed-forward model that reconstructs scenes from unposed images as compact, observation-grounded scene tokens, each derived from a source patch and decoded into freely placed Gaussians. The reconstruction budget is adjustable at inference without retraining.
- We introduce two geometric losses that ground and shape this representation geometrically: a cone loss that ties each token's decoded Gaussians to its source patch, and GeoCE, which aligns token similarity with 3D proximity for cross-view clustering.
- We introduce TokenBA, an efficient token-level bundle adjustment that exploits the resulting 2D–3D correspondences to jointly refine scene tokens and camera poses through reprojection and photometric consistency within a few seconds.

2 Related Work

Feed-forward 3D Gaussian Splatting. Per-scene optimization of radiance fields (Mildenhall et al., 2020; Barron et al., 2022) and 3D Gaussians (Kerbl et al., 2023) produces high-quality novel views but is costly. Feed-forward methods instead predict Gaussians in a single pass, using epipolar attention, cost volumes, or scalable transformers (Charatan et al., 2024; Chen et al., 2024; Xu et al., 2025; Zhang

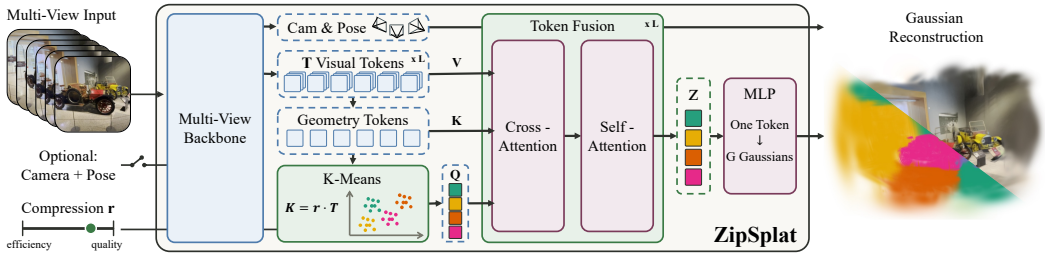


Figure 2: Overview of ZipSplat. Given N unposed images, a multi-view backbone extracts per-patch geometry tokens that are clustered into K scene tokens using k-means. Each scene token is derived from its source patches, and cross- and self-attention refine them. An MLP finally decodes each into G spatially clustered Gaussians (colors denote tokens). The number of scene tokens K is set by the compression ratio r at inference, giving a single model a continuous quality–efficiency trade-off.

et al., 2024a; Ziwen et al., 2025). Pose-free variants progressively remove camera assumptions (Ye et al., 2025a; Smart et al., 2024; Hong et al., 2025; Huang & Mikolajczyk, 2025; Kang et al., 2025), with recent methods jointly estimating cameras and Gaussians from fully uncalibrated inputs (Jiang et al., 2025; Zhang et al., 2025; Ye et al., 2026; Lin et al., 2026). Most, however, retain a pixel-to-Gaussian mapping: this grounds each prediction but makes the Gaussian count scale with the input rather than scene complexity. Pruning, merging, and sparse pixel selection reduce this set only after per-view prediction (Ziwen et al., 2025; Jiang et al., 2025; Wang et al., 2024; Zhang et al., 2024b; Wang et al., 2026; Park et al., 2026; Zhang et al., 2026; Moreau et al., 2026; Fei et al., 2024; Kang et al., 2026), while input compression (Wang et al., 2025b) and storage codecs (Chen et al., 2025; Song et al., 2026; Lee et al., 2024; Morgenstern et al., 2024) do not change this representation.

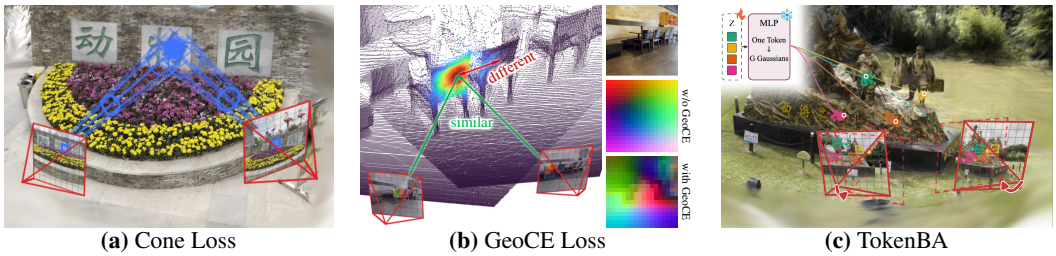
Token-based Gaussian prediction. Token-based methods remove pixel alignment by decoding freely placed Gaussians from latent tokens, similar to Scaffold-GS (Lu et al., 2024), which decodes them from per-scene anchors. TokenGS (Ren et al., 2026), C3G (An et al., 2026), and GlobalSplat (Itkin et al., 2026) aggregate multi-view features with learned queries, while TokenSplat (Li et al., 2026) voxelizes and merges image tokens before decoding. These approaches fuse repeated observations and decouple the Gaussian count from input size, but their query count or voxelization is fixed, so changing the budget requires retraining. Learned queries also lack explicit supporting observations. In contrast, ZipSplat clusters image tokens in a learned geometric space, preserving their observation support while allowing the budget to be adjusted at inference through the clustering ratio.

Post-hoc Refinements. Learned reconstruction systems commonly end with a refinement step: DROID-SLAM (Teed & Deng, 2021) embeds a dense bundle adjustment layer in its update, and VGGT (Wang et al., 2025a) refines its predicted cameras by bundle adjustment over predicted tracks. Feed-forward 3DGS methods so far ignore or lack such correspondences, so refinement falls back to photometric optimization with ground-truth poses. C3G (An et al., 2026) optimizes the decoded Gaussians as in per-scene 3DGS, while TokenGS (Ren et al., 2026) tunes its learnable queries through the frozen network, which preserves geometry but trails Gaussian-space tuning in quality. TokenBA instead uses the built-in 2D–3D correspondences of scene tokens to refine the scene representation and camera poses, achieving faster convergence and higher quality without pose supervision.

3 Method

Given N unposed and uncalibrated images $I = \{I_n \in \mathbb{R}^{H\times W\times 3}\}_{n=1}^{N}$, ZipSplat predicts a set of 3D Gaussians (Kerbl et al., 2023) and estimates the camera parameters. Each Gaussian is parameterized by a center $\boldsymbol{\mu} \in \mathbb{R}^3$, scales $\mathbf{s} \in \mathbb{R}^3$, rotation quaternion $\mathbf{q} \in \mathbb{R}^4$, opacity $\alpha \in [0, 1]$, and color coefficients $\mathbf{c} \in \mathbb{R}^C$. Novel views are rendered via differentiable rasterization (Ye et al., 2025b).

Overview. Figure 2 summarizes the pipeline. A multi-view backbone produces geometry tokens from per-patch features. These are clustered into scene tokens, refined through cross- and self-attention, and decoded into freely placed 3D Gaussians, while camera parameters are estimated alongside. We derive each scene token from its source patches, and a cone loss grounds its Gaussians near them, which gives persistent 2D–3D correspondences (Section 3.1). Clustering removes cross-view redundancy at an adjustable budget, guided by GeoCE (Section 3.2), and TokenBA refines cameras and tokens via the patch–Gaussian associations (Section 3.3).



(a) Cone Loss **(b)** GeoCE Loss **(c)** TokenBA

Figure 3: Grounding, shaping, and refining scene tokens. (a) The cone loss ties each token's Gaussians back to its source patch, grounding the decoded geometry in the observation that produced it. (b) GeoCE aligns token similarity with 3D proximity so that clustering merges patches observing the same surface. (c) TokenBA refines cameras via robust PnP on the patch–Gaussian correspondences and updates scene tokens through the frozen decoder using photometric error on the input views.

3.1 Observation-Grounded Scene Tokens

Each image patch observes a specific surface region. To give each observation a dedicated token, a multi-view backbone extracts per-patch visual tokens at multiple layers. These are merged into geometry tokens $\mathbf{g}$, one per image patch, forming a shared space for clustering, attention, and geometric supervision. Each geometry token serves as a scene token.

Gaussian decoding. Each scene token represents a surface region whose spatial extent and complexity vary across the scene. An MLP decoder D maps each scene token $\mathbf{z}_k$ into G freely placed Gaussians:

$$\mathcal{G}_k = \left\{ \left(\boldsymbol{\mu}_{kj}, \mathbf{s}_{kj}, \mathbf{q}_{kj}, \alpha_{kj}, \mathbf{c}_{kj} \right) \right\}_{j=1}^{G} = D(\mathbf{z}_k), \tag{1}$$

with the full scene assembled from all K tokens. Free placement lets the token spread its Gaussians over the observed region rather than anchoring each one to a fixed ray. However, nothing constrains decoded Gaussians to stay near their source patch: the decoder can place them anywhere in 3D, and the association between token and observation is not guaranteed.

Cone loss. The cone loss grounds each token's Gaussians in its source patch (Fig. 3a). For each token k, it penalizes the reprojection distance between its Gaussian centers and the source patch center:

$$\mathcal{L}_{\text{cone}} = \sum_{k=1}^{K} \sum_{j=1}^{G} \rho\left(\left\| \pi_k(\boldsymbol{\mu}_{kj}) - \mathbf{u}_k \right\| \right), \tag{2}$$

where π_k projects into the source view of token k, $\mathbf{u}_k$ is its patch center, distances are measured in patch units, and ρ is a robust penalty that limits the influence of large residuals. This constraint is only definable because each token has a known source patch. A frustum loss, as used by TokenGS (Ren et al., 2026), keeps Gaussians visible but cannot tie them to a specific observation. With the cone loss, our approach yields an observation-grounded scene representation: each scene token is derived from a source patch, its Gaussians are tied back to that patch, and these associations persist at inference.

3.2 Merging Redundant Scene Tokens

When multiple views cover a scene, many patches observe the same surface from different viewpoints. Their tokens carry redundant information and can be merged into a smaller set whose size is adjustable at inference without retraining.

Inference-time clustering. We cluster the geometry tokens into K groups using k-means in cosine space. Given T patches per view, the budget K is set through the compression ratio

$$r = \frac{K}{NT}, \qquad r \in (0, 1]. \tag{3}$$

Cluster initialization favors photometrically complex regions to preserve fine detail. Each scene token is initialized as the mean of its cluster members, $\mathbf{z}_k^0 = |C_k|^{-1} \sum_{i \in C_k} \mathbf{g}_i$, and retains them as supporting observations. Assignments remain discrete, while gradients propagate through the cluster means during training. A merged scene token inherits the patches of all its members, and the cone loss takes the minimum over them, $\min_{i \in C_k} \|\pi_i(\boldsymbol{\mu}_{kj}) - \mathbf{u}_i\|$, so each Gaussian must fall into the



full capacity $5\times$ compression $20\times$ compression

Figure 4: One model, any budget. The same scene at three compression ratios r, set at inference without retraining. Each pair shows the rendered view (left) and Gaussians colored by their scene token (right). With more compression, fewer tokens need to cover larger regions but the render stays sharp.

view cone of at least one member patch. The free region is the union of the member cones and grows with the cluster. Figure 4 shows the effect of varying r.

Token fusion. To recover detail lost during cluster averaging, we refine the scene tokens through alternating cross- and self-attention:

$$\mathbf{z}^l = \mathrm{SA}\big(\mathrm{CA}(\mathbf{z}^{l-1}, \mathbf{g}, \mathbf{f}_l)\big), \qquad l = 1, \ldots, L, \tag{4}$$

where the geometry tokens $\mathbf{g}$ serve as keys, biasing retrieval toward relevant observations across views, and the backbone features $\mathbf{f}_l$ provide the values, processed from deep to shallow layers. A final residual cross-attention incorporates color features:

$$\mathbf{z} = \mathbf{z}^L + \mathrm{CA}(\mathbf{z}^L, \mathbf{g}, \mathbf{f}^{\mathrm{col}}). \tag{5}$$

GeoCE loss. Clustering can merge tokens across views, but nothing guarantees it groups patches that observe the same surface. GeoCE makes this explicit by shaping the token space (Fig. 3b). During training, each token's patch is unprojected to a 3D anchor $\mathbf{p}_i$ using its median depth. From these anchors we construct a soft target distribution over token pairs, weighted by 3D proximity:

$$w_{ij} \propto \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\sigma_i\sigma_j}\right), \qquad j \neq i, \tag{6}$$

where σ_i scales with the local patch spacing (Section A.2). Nearby patches get high weight; distant ones get near zero. GeoCE pushes the token cosine similarities to match this target via cross-entropy:

$$\mathcal{L}_{\mathrm{geo}} = -\sum_i \sum_{j \neq i} w_{ij} \log \frac{\exp(\mathbf{g}_i^\top \mathbf{g}_j/\tau)}{\sum_{k \neq i} \exp(\mathbf{g}_i^\top \mathbf{g}_k/\tau)}, \tag{7}$$

with temperature τ. Since the target is soft, multiple patches can share high weight when they observe the same surface, making cross-view merging explicit rather than leaving it to the rendering loss.

Training. The model is trained end-to-end with rendering, geometric, and camera supervision:

$$\mathcal{L} = \mathcal{L}_{\mathrm{render}} + \lambda_{\mathrm{geo}}\,\mathcal{L}_{\mathrm{geo}} + \lambda_{\mathrm{cone}}\,\mathcal{L}_{\mathrm{cone}} + \lambda_{\mathrm{pose}}\,\mathcal{L}_{\mathrm{pose}}. \tag{8}$$

The rendering loss combines ℓ_1 and perceptual terms on novel target views, while $\mathcal{L}_{\mathrm{pose}}$ supervises the predicted cameras. Coefficients and details are in Section A.4.

3.3 TokenBA: Refining Cameras and Scene Tokens

The feed-forward prediction carries residual camera and geometry error. The patch–Gaussian associations from Section 3.1 persist at inference and provide 2D–3D correspondences for refining both cameras and scene tokens from the input views within seconds.

Camera update. With the geometry fixed, we update each camera pose $\mathbf{P}_n$ using the Gaussian–patch correspondences. For each Gaussian j of token k with a supporting patch $i \in C_k$ visible in view n, we minimize the robust reprojection error:

$$E_{\mathrm{cam}}(\mathbf{P}_n) = \sum_{(k,j,i)} \rho\big(\|\pi(\mathbf{P}_n, \boldsymbol{\mu}_{kj}) - \mathbf{u}_i\|\big), \tag{9}$$

where ρ is the penalty from Eq. (2). Each pose is estimated independently by robust PnP, minimizing Eq. (9) over all its correspondences with graduated non-convexity, followed by a short photometric polish to reduce patch-center quantization.



Figure 5: Qualitative comparison on DL3DV from 16 to 128 input views. ZipSplat maintains sharp, detailed renders as input coverage increases, outperforming both the token-based TokenGS and the pixel-aligned YoNoSplat and DA3. TokenBA (tBA) refines the reconstruction, most visibly on fine detail like text and textures.

Scene token update. With the cameras fixed, we refine the scene tokens by minimizing the rendering loss on the input views. Gradients propagate through the renderer and frozen decoder to the tokens, combining evidence from all views. Optimizing tokens instead of raw Gaussians retains the decoder as a learned scene prior.

Alternating optimization. Camera correction improves the consistency of rendering gradients, while token refinement improves the geometry for subsequent camera updates. TokenBA alternates these steps and concludes with a short joint photometric refinement of tokens and poses, improving the feed-forward reconstruction within a few seconds (Fig. 3c).

4 Experiments

Implementation Details. We initialize the backbone and camera head from DA3-Giant (Lin et al., 2026). Features from three backbone layers are merged into geometry tokens and fused into scene tokens through three cross- and self-attention blocks. Each scene token produces $G{=}32$ Gaussians with degree-1 spherical harmonics. We train on an equal mixture of RealEstate10K (Zhou et al., 2018) and DL3DV (Ling et al., 2024) at $252{\times}252$ resolution for 450K steps on 16 GH200 GPUs. The input-view count increases from 2 to 24 over the first half of training. Rendering supervision combines ℓ_1 and LPIPS (Zhang et al., 2018) (weight 0.1) on four target views, with auxiliary weights $\lambda_{\text{geo}}{=}0.05$, $\lambda_{\text{cone}}{=}0.01$, and $\lambda_{\text{pose}}{=}0.05$. We sample the compression ratio uniformly from $[0.2/N, 1]$, giving a minimum budget of approximately 2K Gaussians. TokenBA uses up to four alternating rounds followed by at most 96 joint refinement steps with an ℓ_1 objective, adding approximately 1.6 s per scene at 16 views on an RTX 4090. Further details are provided in Section A.

4.1 Multi-View Novel View Synthesis

Setup. We evaluate on DL3DV (Ling et al., 2024) with 16 and 64 input views, and on RealEstate10K (Zhou et al., 2018) with 6 input views. Target views are held out across the full sequence, and context views are chosen from the remaining frames to maximize coverage (Section B.1). Each method runs inference at their training resolution, and all renderings are evaluated at

Table 1: Novel view synthesis on DL3DV (16 and 64 views) and RealEstate10K (6 views). Posed methods receive ground-truth cameras and intrinsics, others estimate them from images. Best, second, third.

Method	Posed	DL3DV 16v				DL3DV 64v				RE10K 6v			
		#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓
DepthSplat (Xu et al., 2025)	✓	1.8M	19.64	0.660	0.278	7.3M	17.32	0.555	0.390	688K	24.62	0.873	0.138
ZPressor (Wang et al., 2025b)	✓	688K	18.44	0.575	0.339	688K	18.36	0.568	0.345	688K	24.89	0.864	0.141
VolSplat (Wang et al., 2026)	✓	1.0M	15.42	0.385	0.565	4.2M	15.55	0.398	0.566	393K	22.98	0.807	0.230
GlobalSplat (Itkin et al., 2026)	✓	16K	15.48	0.397	0.630	16K	15.72	0.402	0.626	16K	23.68	0.782	0.232
TokenGS (Ren et al., 2026)	✓	262K	21.41	0.695	0.345	262K	22.66	0.740	0.328	262K	27.59	**0.899**	0.145
AnySplat (Jiang et al., 2025)		2.3M	18.40	0.559	0.314	7.2M	19.91	0.617	0.278	775K	22.60	0.791	0.189
EcoSplat (Park et al., 2026)		419K	14.05	0.319	0.605	1.7M	13.06	0.275	0.647	157K	21.73	0.728	0.267
C3G (An et al., 2026)		2K	12.71	0.286	0.649	2K	11.56	0.243	0.672	2K	20.64	0.663	0.318
DA3 (Lin et al., 2026)		4.1M	20.34	0.652	0.276	16.3M	20.59	0.666	0.257	1.5M	21.37	0.726	0.229
YoNoSplat (Ye et al., 2026)		803K	20.36	0.638	0.299	3.2M	21.00	0.674	0.275	301K	25.05	0.830	0.154
ZipSplat		166K	**24.79**	**0.808**	**0.175**	664K	**28.99**	**0.900**	**0.106**	62K	**28.50**	0.882	**0.120**

Table 2: Cross-dataset generalization to Mip-NeRF360 and ScanNet++. Posed methods receive ground-truth cameras and intrinsics, others estimate them from images. Best, second, third. *: trained on ScanNet++.

	Method	Posed	32v				64v				128v			
			#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓
Mip-NeRF	$3DGS_{30k}$ (Kerbl et al., 2023)	✓	1.0M	21.28	0.678	0.282	1.2M	24.84	0.806	0.188	1.3M	27.21	0.862	0.137
	AnySplat (Jiang et al., 2025)		4.5M	19.10	0.537	0.322	8.2M	19.91	0.551	0.302	14.5M	20.03	0.556	0.302
	DA3 (Lin et al., 2026)		8.1M	21.48	0.603	0.266	16.3M	21.03	0.585	0.277	32.5M	20.62	0.579	0.287
	TokenGS (Ren et al., 2026)	✓	262K	22.84	0.648	0.434	262K	23.07	0.656	0.432	262K	23.22	0.663	0.424
	YoNoSplat (Ye et al., 2026)		1.6M	17.61	0.411	0.484	3.2M	17.73	0.417	0.479	6.4M	17.54	0.416	0.494
	ZipSplat		332K	**24.43**	**0.696**	**0.210**	664K	**25.24**	**0.723**	**0.185**	1.3M	**25.81**	**0.739**	**0.169**
ScanNet++	$3DGS_{30k}$ (Kerbl et al., 2023)	✓	1.2M	20.11	0.690	0.365	1.6M	21.16	0.732	0.345	1.7M	22.14	0.775	0.324
	AnySplat* (Jiang et al., 2025)		3.5M	16.63	0.669	0.391	6.9M	18.36	0.694	0.348	12.8M	18.53	0.692	0.354
	DA3* (Lin et al., 2026)		8.1M	18.48	0.683	0.394	16.3M	19.15	0.702	0.362	32.5M	19.12	0.703	0.361
	TokenGS (Ren et al., 2026)	✓	262K	22.14	0.772	0.422	262K	22.49	0.782	0.426	262K	22.71	0.786	0.427
	YoNoSplat (Ye et al., 2026)		1.6M	16.13	0.622	0.516	3.2M	16.29	0.626	0.506	6.4M	16.12	0.623	0.513
	ZipSplat		332K	**22.36**	**0.779**	**0.298**	664K	**24.36**	**0.826**	**0.249**	1.3M	**25.40**	**0.848**	**0.235**

252×252 (Section B.4). All results use full capacity (r=1). Each method runs its full released inference setting: TokenGS applies test-time token optimization (TTO) and ZipSplat applies TokenBA.

Baselines. We compare against pose-free methods AnySplat (Jiang et al., 2025), EcoSplat (Park et al., 2026), DA3 (Lin et al., 2026), C3G (An et al., 2026), and YoNoSplat (Ye et al., 2026). Baselines using ground-truth cameras include the pixel-aligned DepthSplat (Xu et al., 2025) and ZPressor (Wang et al., 2025b), voxel-based VolSplat (Wang et al., 2026), and token-based GlobalSplat (Itkin et al., 2026) and TokenGS (Ren et al., 2026). DA3 uses the same backbone as ZipSplat, providing a comparison between pixel-aligned prediction and our scene-token formulation with a shared backbone.

Results. Table 1 shows that ZipSplat achieves the highest quality in all three settings, including against methods given ground-truth cameras. On DL3DV, the advantage grows with more views: from 4.45 dB over DA3, which shares our backbone, at 16 views to 8.40 dB at 64 views, with about 24× fewer Gaussians. Pixel-aligned prediction ties each Gaussian to a single view's depth and pose errors; as views increase, these misalignments accumulate rather than cancel, whereas our free 3D placement avoids this dependence (Fig. 5). Against TokenGS with TTO, the strongest token-based baseline, the gap widens from 3.38 to 6.33 dB across the same range. Despite its strong PSNR, TokenGS blurs fine details and misses scene content (Figs. 5 and 8), because learned queries must infer their observation support implicitly. Our scene tokens start from their supporting patches, preserving content and producing sharper reconstructions. On RealEstate10K, ZipSplat exceeds TokenGS by 0.91 dB with 4× fewer Gaussians; posed methods lead only in SSIM, likely benefiting from ground-truth intrinsics. Even without TokenBA, ZipSplat outperforms all evaluated feed-forward baselines in PSNR and LPIPS on every benchmark (Section F.1).

4.2 Cross-Dataset Generalization

We evaluate zero-shot generalization on Mip-NeRF360 (Barron et al., 2022) and ScanNet++ (Yeshwanth et al., 2023). Mip-NeRF360 contains indoor and outdoor scenes, while ScanNet++ covers cluttered indoor rooms. We follow the sampling protocol used in Section 4.1.

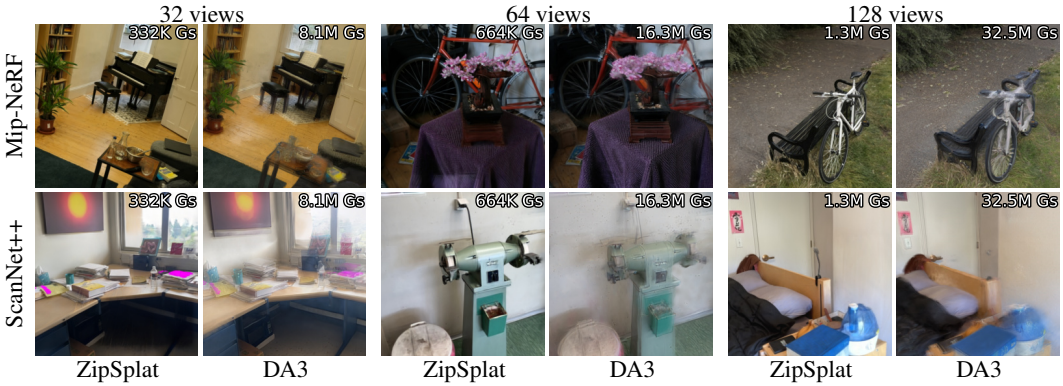


Figure 6: Cross-dataset qualitative comparison on Mip-NeRF360 (top) and ScanNet++ (bottom) at 32, 64, and 128 input views. For each view count we show ZipSplat (left) and DA3 (right). ZipSplat produces sharper geometry and fewer artifacts, with quality improving steadily as more views become available.

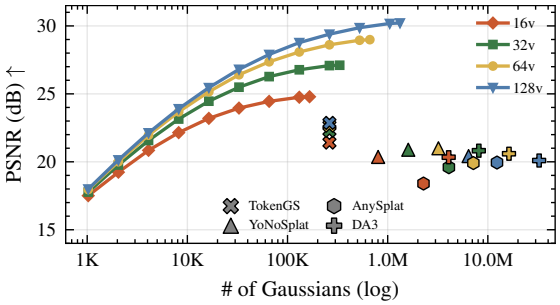


Figure 7: Quality *vs.* Gaussian budget on DL3DV. Each ZipSplat curve varies the compression ratio r at inference. More input views raise the curve at every budget.

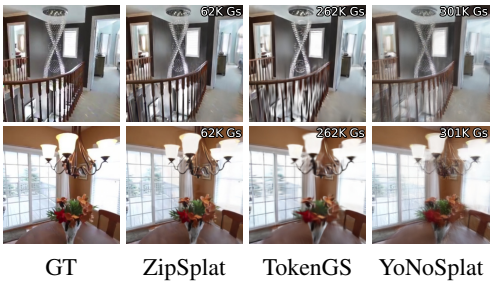


Figure 8: Comparison on RealEstate10K (6 views). ZipSplat reconstructs fine details more faithfully than the baselines with fewer Gaussians.

We compare against the main pose-free methods, AnySplat, DA3, and YoNoSplat, and against TokenGS as the token-based reference. AnySplat and DA3 train on ScanNet++, so their results there are not zero-shot. Per-scene 3DGS (Kerbl et al., 2023) with ground-truth cameras and 30K iterations provides an optimization-based reference.

Table 2 and Fig. 6 show that ZipSplat outperforms every pose-free baseline and TokenGS, which requires ground-truth cameras, across all metrics and view counts on both datasets. Its quality rises with denser input, by 1.38 dB on Mip-NeRF360 and 3.04 dB on ScanNet++ from 32 to 128 views, while the baselines largely plateau or decline. Even without cameras, ZipSplat exceeds per-scene 3DGS on ScanNet++ by 3.26 dB and comes within 1.40 dB of it on Mip-NeRF360 at 128 views.

4.3 Adjustable Gaussian Budget

Since the compression ratio r is an inference-time parameter, one trained model produces reconstructions at any point of the quality–budget curve. Figure 7 traces this curve on DL3DV for 16 to 128 input views, from 1K Gaussians up to full capacity. Quality degrades gracefully: at 16 views, reducing the budget from 166K to 65K costs only 0.34 dB with about 2.5× fewer Gaussians. Compressing further to 8K Gaussians still exceeds every baseline, with 100× fewer Gaussians than YoNoSplat and 500× fewer than DA3 at 16 views. The smallest budget of 1K Gaussians lies below the smallest training budget of about 65 scene tokens, yet the curve continues smoothly. Additional views raise the entire curve rather than only its endpoint. At a fixed budget of 8K Gaussians, PSNR rises from 16 to 128 views by 1.76 dB, so quality follows the observations rather than the number of Gaussians. In Fig. 4, scene tokens cover larger regions as r decreases while the render stays sharp.

4.4 Pose Estimation and TokenBA

Table 3 reports pose accuracy as the AUC of the relative pose error at 1°, 3°, and 5°. ZipSplat achieves the highest accuracy at every threshold on both datasets. Compared with DA3, AUC at 1° increases from 0.930 to 0.950 on DL3DV and from 0.546 to 0.738 on RealEstate10K.

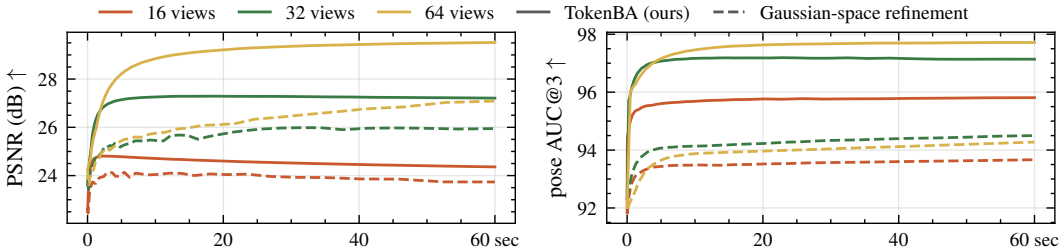


Figure 9: TokenBA *vs.* Gaussian-space refinement on DL3DV. Target-view PSNR (left) and pose accuracy (right) vs. refinement time. TokenBA converges faster and to a higher level than direct Gaussian optimization.

Table 3: Pose estimation. AUC of relative pose error. We color the best and second best.

Method	DL3DV 64v				RE10K 6v		
	AUC↑	1°	3°	5°	1°	3°	5°
AnySplat (Jiang et al., 2025)	0.071	0.357	0.550	0.130	0.378	0.530	
DA3 (Lin et al., 2026)	0.930	0.969	0.980	0.546	0.741	0.817	
YoNoSplat (Ye et al., 2026)	0.466	0.754	0.842	0.361	0.673	0.779	
ZipSplat	**0.950**	**0.976**	**0.983**	**0.738**	**0.855**	**0.897**	

Table 4: Ablations on DL3DV (6 views). From per-pixel to fully grounded scene tokens.

Method	Adj. budget	Cone	GeoCE	TokenBA	r=1 #Gs	r=1 PSNR↑	0.5 #Gs	0.5 PSNR↑	AUC@3↑ r=1
Per-pixel	–	–	–	–	381K	22.62	–	–	0.542
Fixed queries	×	frustum	–	–	262K	23.02	–	–	0.632
Obs. tokens	✓	frustum	–	–	62K	22.87	31K	22.53	0.615
+ Cone	✓	✓	–	–	62K	24.38	31K	23.09	0.669
+ GeoCE	✓	✓	✓	–	62K	24.34	31K	23.25	0.686
+ TokenBA	✓	✓	✓	✓	62K	27.79	31K	27.13	0.889

Figure 9 compares TokenBA against Gaussian-space refinement (Fan et al., 2024) on DL3DV, running both for 60 s without early stopping. TokenBA converges faster and reaches higher quality within this budget: at 16 views, 90% of its gain arrives within 0.4 s, and its margin over Gaussian-space refinement grows from 0.6 dB at 16 views to 2.4 dB at 64. The first camera update alone lifts AUC at 3° from 91.8 to 94.3 within milliseconds, beyond what Gaussian-space refinement reaches in 60 s (93.7). All other results use the early-stopping schedule, which stops after 1.6 to 11.7 s. The token–patch correspondences give TokenBA explicit 2D–3D associations that photometric-only refinement lacks, improving not only convergence speed but also accuracy and robustness.

4.5 Ablations

Table 4 ablates design choices on DL3DV at 6 views; every row is a DA3-Base retrain with the same training budget. Per-pixel prediction gives 22.62 dB from 381K Gaussians with no budget control. Fixed learned queries (matching TokenGS (Ren et al., 2026): 4K queries × 64 Gaussians) reach 23.02 dB at 262K Gaussians with their frustum loss, which constrains Gaussians to lie within the input views, but the budget is locked at training time. Observation-derived tokens make the budget adjustable at inference: 62K Gaussians reach 22.87 dB, and halving to 31K costs only 0.34 dB. Yet the frustum loss does not tie tokens to their source observations. The cone loss grounds them and adds the largest training-time gain (+1.51 dB). It also provides the 2D–3D correspondences TokenBA relies on: without it, TokenBA collapses to 14.7 dB. GeoCE further improves clustering by making tokens that observe the same surface merge, adding +0.16 dB at r=0.5 and lifting AUC@3 from 0.669 to 0.686. With grounding established, TokenBA refines cameras and tokens through the learned correspondences, delivering the large gains in both reconstruction and pose accuracy.

5 Conclusion

We present ZipSplat, a feed-forward model that reconstructs scenes from unposed images as compact, observation-grounded scene tokens. Each token is derived from an image patch and decoded into freely placed 3D Gaussians. Tokens across views are merged, removing redundancy and making the budget adjustable. The cone loss ties each token's Gaussians to its source patch, which is what makes the grounding pay off. GeoCE shapes the token space so that merged tokens observe the same surface. The resulting correspondences enable TokenBA, which refines cameras and tokens within seconds. Without poses or intrinsics, ZipSplat outperforms baselines with substantially fewer Gaussians, including those given ground-truth cameras, and generalizes zero-shot to unseen datasets.

Limitations: The camera estimates can degrade on scenes that differ strongly from the training distribution. Ground-truth cameras close this gap, and TokenBA recovers most of it, but a residual remains on out-of-distribution captures. The method also assumes static scenes.

AI USE STATEMENT

We used generative AI tools to assist with implementing methods and experiments, interpreting results, and drafting and editing the paper. All ideas behind the method are our own, and we have not used generative AI tools to generate data. We have reviewed and verified all AI-assisted work, and all reported numbers were reproduced with our own evaluation runs. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 4 and Section A give the implementation details of ZipSplat, including the architecture, the losses, the training schedule, and the TokenBA settings. Training and evaluation use only public datasets, and Section B documents the evaluation protocol, including the view selection (Section B.1). We re-evaluate all baselines under this same protocol, with the roster in Section B.3 and the full results in Section F. We will release the code and the trained models.

REFERENCES

Honggyu An, Jaewoo Jung, Mungyeom Kim, Chaehyun Kim, Minkyeong Jeon, Jisang Han, Kazumi Fukuda, Takuya Narihira, Hyunah Ko, Junsu Kim, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. Learning compact 3D representations from feed-forward novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 363–373, 2026. 2, 3, 7, 16, 23, 24, 25

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. 2, 7

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian Splats from image pairs for scalable generalizable 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19457–19467, 2024. 1, 2

Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feedforward 3D Gaussian Splatting compression. In *International Conference on Learning Representations*, 2025. 3

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*, pp. 370–386. Springer, 2024. 1, 2

Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, et al. InstantSplat: Sparse-view Gaussian Splatting in seconds. *arXiv preprint arXiv:2403.20309*, 2024. 9, 18

Xin Fei, Wenzhao Zheng, Yueqi Duan, Wei Zhan, Masayoshi Tomizuka, Kurt Keutzer, and Jiwen Lu. PixelGaussian: Generalizable 3D Gaussian reconstruction from arbitrary views. *arXiv preprint arXiv:2410.18979*, 2024. 3

Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jisang Han, Jiaolong Yang, Chong Luo, and Seungryong Kim. PF3plat: Pose-free feed-forward 3D Gaussian Splatting for novel view synthesis. In *International Conference on Machine Learning*, volume 267, pp. 23662–23681, 2025. 3

Ranran Huang and Krystian Mikolajczyk. No pose at all: Self-supervised pose-free 3D Gaussian Splatting from sparse views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27947–27957, 2025. 3

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. 2, 3, 7, 16, 23, 24, 25

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*, 44(6):1–16, 2025. 2, 3, 7, 9, 16, 17, 23, 24, 25

Gyeongjin Kang, Jisang Yoo, Jihyeon Park, Seungtae Nam, Hyeonsoo Im, Sangheon Shin, Sangpil Kim, and Eunbyung Park. SelfSplat: Pose-free and 3D prior-free generalizable 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22012–22022, 2025. 3

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

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023. 1, 2, 3, 7, 8, 16, 24, 25

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

Yihui Li, Chengxin Lv, Zichen Tang, Hongyu Yang, and Di Huang. TokenSplat: Token-aligned 3D Gaussian Splatting for feed-forward pose-free reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 40886–40895, 2026. 3, 16

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. In *International Conference on Learning Representations*, volume 2026, pp. 141261–141285, 2026. 3, 6, 7, 9, 14, 16, 23, 24, 25

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. 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. 6, 15

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. 3

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pp. 405–421, 2020. 1, 2

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. 3

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D scene representation via self-organizing Gaussian grids. In *European Conference on Computer Vision*, pp. 18–34, 2024. 3

Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez, Jaeho Moon, Jihyong Oh, and Munchurl Kim. EcoSplat: Efficiency-controllable feed-forward 3D Gaussian Splatting from multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26010–26020, 2026. 3, 7, 16, 23, 24, 25

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. 2, 3, 4, 7, 9, 16, 18, 23, 24, 25

Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3R: Zero-shot Gaussian Splatting from uncalibrated image pairs. *arXiv preprint arXiv:2408.13912*, 2024. 3

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

Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In *Advances in Neural Information Processing Systems*, volume 34, pp. 16558–16569, 2021. 3

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5294–5306, 2025a. 3

Weijie Wang, Donny Y. Chen, Zeyu Zhang, Duochao Shi, Akide Liu, and Bohan Zhuang. ZPressor: Bottleneck-aware compression for scalable feed-forward 3DGS. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113407–113436, 2025b. 3, 7, 16, 23, 24, 25

Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Feng Chen, Jia-Wang Bian, Zheng Zhu, Donny Y. Chen, and Bohan Zhuang. VolSplat: Rethinking feed-forward 3D Gaussian Splatting with voxel-aligned prediction. In *European Conference on Computer Vision*, 2026. 3, 7, 16, 23, 24

Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. FreeSplat: Generalizable 3D Gaussian Splatting towards free view synthesis of indoor scenes. In *Advances in Neural Information Processing Systems*, volume 37, pp. 107326–107349, 2024. 3

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. 1, 2, 7, 16, 23, 24, 25

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, 2025a. 2, 3, 16, 24

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. 2, 3, 7, 9, 16, 17, 18, 22, 23, 24, 25

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for Gaussian Splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025b. 3

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12–22, 2023. 7

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, 2024a. 2

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/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. 6

Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. FLARE: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21936–21947, 2025. 3

Shengjun Zhang, Xin Fei, Fangfu Liu, Haixu Song, and Yueqi Duan. Gaussian Graph Network: Learning efficient and generalizable Gaussian representations from multi-view images. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50361–50380, 2024b. 3

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, 2026. 3, 16

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):1–12, 2018. 6, 15

Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-LRM: Long-sequence large reconstruction model for wide-coverage Gaussian Splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4349–4359, 2025. 3

A Additional Implementation Details

A.1 Architecture

Backbone features. The backbone is a pretrained DA3-Giant (Lin et al., 2026) (40 layers, width 1536, 24 heads of dimension 64). We extract features from layers 19, 29, and 39, each comprising a local component from within-view attention and a global component incorporating cross-view information.

Geometry tokens. Clustering requires a space where proximity reflects shared 3D structure. The backbone produces per-patch features at multiple layers, each with a local component $\mathbf{f}_l^{\text{loc}}$ from within-view attention and a global component $\mathbf{f}_l^{\text{glob}}$ from cross-view attention. These are fused per layer as

$$\mathbf{f}_l = \mathbf{f}_l^{\text{glob}} + \alpha_l \cdot \text{LN}(\mathbf{f}_l^{\text{loc}}), \tag{10}$$

where α_l is a learnable gate initialized to 0.1. A separate two-layer MLP embeds each raw 14×14 RGB patch into a 1536-dimensional color token $\mathbf{f}^{\text{col}}$. All streams merge into a single geometry token per patch,

$$\mathbf{g} = \sum_l \beta_l \cdot \text{MLP}_l\big(\text{LN}(\mathbf{f}_l)\big) + \beta_c \cdot \text{MLP}_c\big(\text{LN}(\mathbf{f}^{\text{col}})\big), \tag{11}$$

with learnable scales β_l, β_c initialized to $1/4$ and two-layer GELU MLPs. This shared space defines the clustering metric, the cross-attention keys, and the scene token initialization. GeoCE supervises these tokens directly; the per-layer payloads and color embedding supply the attention values.

Fusion and decoding. Fusion uses three blocks. Each block is a cross-attention (queries: scene tokens; keys: geometry tokens; values: the visual tokens of one layer, deepest first) followed by self-attention among the scene tokens. Blocks use 24 heads, MLP ratio 4, GELU, and residual gates initialized to 0.01. A final residual cross-attention with the color embedding as values adds appearance. A two-layer MLP decodes each scene token into G=32 Gaussians of 23 parameters.

Reference frame. All geometry is expressed in the coordinate frame of the first input camera. The scene scale is normalized so that the median distance of the input depth points to the origin is one.

Camera estimation and priors. A two-layer ReLU MLP maps the deepest backbone camera token to a translation, unit quaternion, and field of view, initialized from the DA3 camera head. $\mathcal{L}_{\text{pose}}$ applies a Huber loss on translation, an ℓ_1 chordal loss on rotations, and an ℓ_1 loss on the field of view. When ground-truth cameras are available, DA3's frozen camera encoder injects them into the camera token; this path is used with probability 0.2 during training.

Gaussian parameterization. Positions use an inverse-log activation $\text{sign}(y)(\exp|y| - 1)$ on raw outputs clamped to $[-5, 5]$, scales use $\text{softplus}(x-4)$ clamped to $[10^{-6}, 15]$, rotations are normalized quaternions, opacities use a sigmoid with bias -1.5, and colors are degree-1 spherical harmonics (12 coefficients). All 32 Gaussians of a token start from a shared template with a forward bias of z=0.5. Squared-hinge penalties keep scales in $[10^{-6}, 0.1]$, opacities above 0.01, and positions within a radius of 10. Table 5 varies the number of Gaussians per scene token. G=32 gains 0.34 to 0.36 dB over G=8, while G=64 adds at most 0.08 dB at twice the Gaussians, so ZipSplat uses G=32.

Table 5: Gaussians per scene token on DL3DV at 6 views. Each row is a DA3-Base model retrained to decode G Gaussians per scene token, evaluated without TokenBA. Columns compare full capacity (r=1) with compression ratio r=0.5, where half of the scene tokens are kept. #Gs is the total number of Gaussians.

G	r=1		r=0.5	
	#Gs	PSNR↑	#Gs	PSNR↑
8	16K	23.98	8K	22.91
16	31K	24.20	16K	23.11
32	62K	24.34	31K	23.25
64	124K	24.42	62K	23.33

A.2 Geometric Losses

GeoCE loss. During training, 2048 tokens per scene are sampled as anchors, with each anchor's 3D point obtained by unprojecting its patch's median-depth pixel. The soft target weight of Eq. (6) is

averaged over a 4×4 pixel grid inside each key patch, and patches with valid depth on fewer than half their pixels are excluded. The base bandwidth σ is the scene-wise median 3D distance between adjacent patch points. Per token, σ_i is log-interpolated from σ (flattest patch) to $0.4\,\sigma$ (most textured), ranked by mean image-gradient energy. The temperature is τ=0.1 and the loss acts on ℓ_2-normalized geometry tokens with weight λ_{geo}=0.05. Anchor depths are precomputed by DA3-Giant from the ground-truth cameras of each training scene.

Cone loss. The robust penalty ρ in Eq. (2) is a hinged Cauchy function of the reprojection distance in patch units. The hinge sets a zero-cost region within 2 patches, allowing Gaussians to extend past their observations to cover flat surfaces or occluded regions. Beyond the hinge, the penalty ramps smoothly (C^1) and saturates past 4 patches to limit the influence of distant outliers. Gaussians that project behind a member view receive a sentinel distance of 20 patches with no lateral gradient. The loss constrains only the lateral position relative to the supporting patches' viewing cones and leaves depth to the rendering loss, with weight λ_{cone}=0.01.

A.3 Clustering

We run spherical k-means on the ℓ_2-normalized geometry tokens with cosine assignments and centroid renormalization after each update. Lloyd iterations are capped at 50 but typically converge in 2 to 3 iterations at r=0.5 and 7 to 9 at r=0.1. Seeds are sampled without replacement with probability proportional to $(0.05 + c_i)^2$, where $c_i \in [0, 1]$ is the per-scene rank of patch i by mean image-gradient energy. This biases initialization toward textured regions but does not change the objective. Each scene token is the mean of its members' geometry tokens, recomputed differentiably from the detached assignments. During training, $r \sim \mathcal{U}[r_{\min}, 1]$ with $r_{\min}$=$0.2/N$, giving at least about 65 scene tokens at any view count. A fixed Gaussian budget B corresponds to $r = B/(GNT)$.

A.4 Training

Training is a single stage from DA3-Giant on a 50/50 mixture of RealEstate10K (Zhou et al., 2018) and DL3DV (Ling et al., 2024) at 252×252. It runs for 450K steps on 16 GH200 GPUs with 24 samples per GPU and 4 target views per sample. We use AdamW (β_1=0.9, β_2=0.95, weight decay 0.05, gradient clipping 1.0) with a base learning rate of 3×10^{-4}, the backbone at $0.1\times$, 5% linear warmup, cosine decay to zero, and bfloat16 autocast. The input view count grows from 2 to 24 over the first half of training. A single checkpoint, trained on the RealEstate10K and DL3DV mixture, is used across all benchmarks for evaluation.

A.5 TokenBA

TokenBA optimizes the scene tokens $\mathbf{Z}$ and the input camera poses under an ℓ_1 rendering loss on the input views, with the decoder frozen. Tokens use Adam with a learning rate of 10^{-3} during the alternation and 2×10^{-3} in the joint tail. The decoder MLP body runs in half precision while its readout stays in full precision.

Camera update. The correspondences in Eq. (9) follow from the cluster assignments: every Gaussian of a scene token is paired with the patch center of each member patch. Each pose is solved by robust PnP with graduated non-convexity, annealing the robust scale from 64 to 12 to 4 pixels over 4 iterations per level on 2048 sampled correspondences. Seven rotation hypotheses per view (the current rotation and $\pm40^\circ$ about each camera axis) are solved in one batched call, and the hypothesis with the most inliers within 2 patches wins. A photometric polish then removes the patch-center quantization error, running two steps at half resolution and one at full resolution with the aligner described below.

Photometric aligner. The aligner is a direct Levenberg–Marquardt solver on the photometric residual. Each step renders at the current pose and takes a Gauss–Newton update on the brightness-constancy linearization with the analytic image-warp Jacobian, using one render per step and no autograd. The full configuration uses a four-level pyramid ($1/8$, $1/4$, $1/2$, and full resolution) with 12, 10, 8, and 6 steps per level, Gaussian blur at the two coarse levels, and accept-reject steps with a trust-region clamp on the pose twist. A scan over nine log-spaced global scale factors from 0.5 to 2.0 precedes the pyramid, and the best full-resolution pose across levels is kept. Inside TokenBA only the two fine levels run, without verification or scale scan.

Token update and joint refinement. Each alternation round ends with one token refit step. Rounds stop once the PnP snap moves every pose by less than $0.3°$, or after four rounds. The joint tail refines tokens and poses (an $\mathfrak{se}(3)$ increment with learning rate 3×10^{-4}) for at most 96 steps, reusing the same Adam state. It exits once a block of 8 steps improves the loss by less than 2%.

B Evaluation Protocol

We describe the evaluation setup shared by all experiments: view selection, pose metrics, baselines, and the resolution and resize choices that affect reported numbers.

B.1 View selection

On DL3DV (140 scenes), Mip-NeRF360 (6 scenes), and ScanNet++ (46 scenes), every 8th frame is held out as a target ($\approx$40 per scene), and context views are selected from the remaining frames by k-means++ on the 6-D feature vector of camera position and viewing direction. We evaluate at 16, 32, 64, and 128 input views on DL3DV and at 32, 64, and 128 on Mip-NeRF360 and ScanNet++, with targets fixed across view counts, so results at different rungs are directly comparable. On RealEstate10K (1,600 scenes) we use the 6-view protocol of (Ye et al., 2026) with 8 held-out targets.

B.2 Pose accuracy

We report relative rotation error and translation-direction error over all input-view pairs, pooled over all scenes before computing the AUC. Pooling produces a single error distribution that weights each pair equally, which differs from a mean of per-scene AUCs. The main text reports AUC at 1, 3, and 5 degrees, and Section F.5 extends this to 10, 15, 20, and 30 degrees.

B.3 Baselines

We reimplement each baseline into our codebase and validate it against the original code up to floating-point precision. All methods are then evaluated under the same protocol: the same view indices, the same metric resolution, and the same resize filter. Since a predicted scene need not align perfectly with the ground-truth cameras, we align the target cameras to the predicted scene before computing metrics for every method, following prior work (Ye et al., 2025a; 2026). Pose-free methods (AnySplat (Jiang et al., 2025), EcoSplat (Park et al., 2026), DA3 (Lin et al., 2026), C3G (An et al., 2026), YoNoSplat (Ye et al., 2026)) predict cameras from the input images. Posed methods (DepthSplat (Xu et al., 2025), ZPressor (Wang et al., 2025b), VolSplat (Wang et al., 2026), GlobalSplat (Itkin et al., 2026), TokenGS (Ren et al., 2026)) receive ground-truth poses and intrinsics. Per-scene 3DGS (Kerbl et al., 2023) optimizes 100K Gaussians, initialized randomly around the camera centroids, for 30K iterations on the context views with ground-truth cameras, providing an optimization-based reference. Each method's encoder runs at its own training resolution, and all renders are scored at a common 252×252 (Section B.4). SparseSplat (Zhang et al., 2026) and TokenSplat (Li et al., 2026) had no public code at the time of writing.

B.4 Rendering resolution

Methods train at different resolutions and aspect ratios, and scoring at a lower resolution raises PSNR, since downsampling removes the fine detail that is hardest to reconstruct. To compare fairly, we score every method at a common 252×252 while still running each encoder at its own training resolution. Table 6 compares this protocol with scoring each baseline at its own resolution and with running it entirely at 252. Scoring each baseline at its own resolution would favor the low-resolution ones and penalize the high-resolution ones: YoNoSplat gains 0.15 dB at its native 224, while AnySplat loses 0.31 dB at its native 448. Running a baseline entirely at 252 is worse still, costing AnySplat 4.2 to 5.0 dB and YoNoSplat 2.1 to 2.8 dB. Our protocol therefore keeps each encoder at its training resolution, and the high-resolution baselines keep the detail they encode.

Table 6: Effect of evaluation resolution on DL3DV. Methods train at different native resolutions, and the scoring resolution changes the reported metrics. *Src*, *Enc*, and *Eval* denote the resolution for loading, encoding, and scoring. The middle row of each baseline is our main-table protocol: each encoder runs at its native resolution, but all methods are scored at a common 252×252. ZipSplat trains at 252, so all three settings coincide.

Method	Src	Enc	Eval	16v			32v			64v		
				PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
AnySplat (Jiang et al., 2025)	540	448	448	18.09	0.556	0.369	19.42	0.597	0.338	19.83	0.613	0.332
	540	448	252	18.40	0.559	0.314	19.60	0.603	0.283	19.91	0.617	0.278
	252	252	252	14.16	0.296	0.561	14.74	0.307	0.564	14.90	0.317	0.576
YoNoSplat (Ye et al., 2026)	540	224	224	20.51	0.647	0.283	21.03	0.673	0.262	21.10	0.682	0.260
	540	224	252	20.36	0.638	0.299	20.89	0.664	0.278	21.00	0.674	0.275
	252	252	252	18.26	0.492	0.379	18.38	0.496	0.378	18.22	0.495	0.386
ZipSplat w/o TokenBA	540	252	252	22.61	0.725	0.213	23.65	0.763	0.182	24.25	0.786	0.165

Table 7: Effect of resize filter on DL3DV. Each method encodes at its native resolution and is scored at 252×252. The two rows per method compare PIL Lanczos and bilinear with antialiasing as the filter for downsampling the ground-truth images. The filter choice alone changes PSNR, so results are not comparable across papers unless the filter is reported. We use bilinear with antialiasing throughout.

Method	Filter	16v			32v			64v		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
AnySplat (Jiang et al., 2025)	PIL Lanczos	18.02	0.533	0.326	19.11	0.573	0.297	19.36	0.584	0.294
	bilinear+AA	18.40	0.559	0.314	19.60	0.603	0.283	19.91	0.617	0.278
YoNoSplat (Ye et al., 2026)	PIL Lanczos	19.77	0.602	0.319	20.24	0.626	0.299	20.37	0.638	0.296
	bilinear+AA	20.36	0.638	0.299	20.89	0.664	0.278	21.00	0.674	0.275
ZipSplat w/o TokenBA	PIL Lanczos	21.75	0.689	0.226	22.71	0.728	0.195	23.27	0.752	0.177
	bilinear+AA	22.61	0.725	0.213	23.65	0.763	0.182	24.25	0.786	0.165

B.5 Resize filter

Switching the ground-truth downsampling filter from PIL Lanczos to bilinear with antialiasing changes PSNR by up to 0.98 dB (Table 7). Results are therefore not comparable across papers unless the filter is reported. We use bilinear with antialiasing for every method, matching how the DL3DV source ground truth is downscaled.

C TokenBA Analysis

This section compares TokenBA with the test-time refinement of the baselines, breaks down its runtime, ablates its components, and shows qualitative results with and without it.

C.1 Comparison with baseline refinement

Table 8 compares TokenBA with the test-time refinement available to the baselines, ranking feed-forward and refined predictions separately. TokenGS optimizes its scene latents at test time, which lifts it by 1.3 to 2.1 dB PSNR from 16 to 64 views. TokenBA lifts ZipSplat by 2.2 to 4.7 dB over the same range, so the gap to TokenGS widens from 2.5 to 3.7 dB before refinement to 3.4 to 6.3 dB after it, although TokenGS receives ground-truth cameras. TokenBA is also 3.2 to 3.5 times faster, taking 1.8 to 12.8 s per scene including the feed-forward pass against 6.3 to 41.4 s for TokenGS with its optimization. Refining the Gaussians of ZipSplat w/o TokenBA directly for 60 s reaches 23.7 to 27.1 dB, 1.0 to 1.9 dB below TokenBA, which needs only 1.6 to 11.7 s of refinement. TokenBA also refines the cameras, raising AUC at 3° from 0.92 to between 0.95 and 0.98, well above the 0.74 to 0.76 of YoNoSplat before refinement. Gaussian-space refinement also helps the baselines, lifting YoNoSplat by 2.5 dB and TokenGS by 1.8 dB at 16 views, more than TokenGS gains from its own latent optimization, yet both stay 1.9 and 2.8 dB below TokenBA.

Table 8: Test-time refinement on DL3DV. Feed-forward predictions (top) and their refined counterparts (bottom), ranked separately. TokenBA refines both Gaussians and cameras through the frozen decoder. Gaussian-space refinement (GSR) (Fan et al., 2024) optimizes the predicted Gaussians for 60 s per scene. TTO is the test-time optimization of TokenGS. Posed methods receive ground-truth cameras and have no pose metric.

Method	Posed	16v			32v			64v		
		PSNR↑	LPIPS↓	AUC@3↑	PSNR↑	LPIPS↓	AUC@3↑	PSNR↑	LPIPS↓	AUC@3↑
TokenGS (Ren et al., 2026)	✓	20.16	0.433	N/A	20.46	0.428	N/A	20.56	0.427	N/A
YoNoSplat (Ye et al., 2026)		20.36	0.299	0.738	20.89	0.278	0.755	21.00	0.275	0.754
ZipSplat w/o TokenBA		**22.61**	**0.213**	**0.918**	**23.65**	**0.182**	**0.923**	**24.25**	**0.165**	**0.920**
TokenGS + GSR (Ren et al., 2026)	✓	21.98	0.272	N/A	24.17	0.214	N/A	25.54	0.185	N/A
TokenGS + TTO (Ren et al., 2026)	✓	21.41	0.345	N/A	22.28	0.332	N/A	22.66	0.328	N/A
YoNoSplat + GSR (Ye et al., 2026)		22.89	0.255	0.764	25.10	0.190	0.787	25.38	0.180	0.786
ZipSplat w/o TokenBA + GSR		23.74	0.208	0.937	25.99	0.159	0.945	27.10	0.135	0.943
ZipSplat		**24.79**	**0.175**	**0.954**	**27.11**	**0.133**	**0.970**	**28.99**	**0.106**	**0.976**

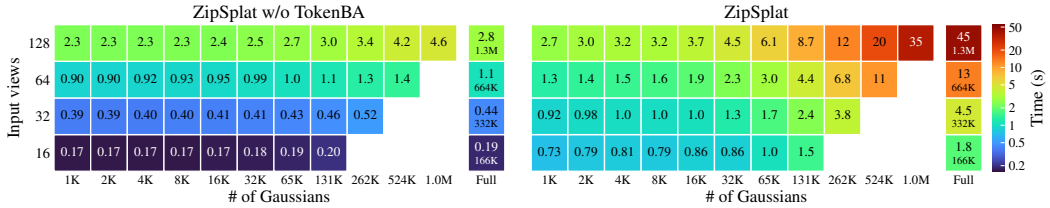


Figure 10: Inference time vs. input views and Gaussian budget on DL3DV, feed-forward (left) and feed-forward + TokenBA (right), in seconds per scene on an RTX 4090. Without TokenBA, the time depends mostly on the number of views. TokenBA adds time that grows with both the number of views and the Gaussian count.

C.2 Timing breakdown

The cost of TokenBA is set by how much it renders. Figure 10 times the grid of view counts and Gaussian budgets with and without TokenBA, and the difference between the two panels is the refinement itself. Times follow the protocol of Section E, and the TokenBA panel averages over all 140 test scenes since its early stops make the cost scene-dependent. This difference grows nearly linearly with both the number of views and the number of Gaussians, since every step renders all input views. At 128 views, each doubling of the budget from 65K upward multiplies the cost of TokenBA by 1.6 to 2.0, and at 262K Gaussians each doubling of the views multiplies it by 1.7. Without TokenBA, the three largest budgets at 128 views are up to 1.7 s slower than full capacity. Full capacity keeps every token and skips clustering. A compressed budget first runs k-means on all 41K tokens of the 128 views. With many clusters, each k-means iteration compares every token with every cluster center, and k-means needs 17 to 45 iterations to converge. This costs 1.9 s at 524K Gaussians. Table 9 breaks down a single step at 16 views and full capacity.

Table 9: TokenBA per-step cost at 16 views and full capacity on an RTX 4090.

Phase	Time (ms)
PnP snap	2.1
Photometric polish	15.0
Token update	18.3
Total	35.5

(a) One alternation round

Phase	Time (ms)
Rasterizer backward	10.8
Rasterizer forward	4.3
Decoder forward and backward	2.4
Adam update	0.4
Total	18.1

(b) One joint tail step

Each alternation round takes 36 ms, while the rasterizer's forward and backward passes take 83% of each tail step. The joint tail exits after 94 of its 96 steps on average at 16 views. Compression is therefore also the lever on the refinement cost. At 128 views, reducing the budget from 1M to 262K Gaussians cuts TokenBA from 31 s to 9.0 s, at a cost of 0.8 dB (Fig. 14).

C.3 Ablations

Table 10 removes one component of TokenBA at a time, in domain on DL3DV at 16 views and out of domain on Mip-NeRF 360 at 32 views. Updating the poses matters most. Refitting the tokens

Table 10: TokenBA ablations. Each row removes one component of the full TokenBA. Time is per scene and includes the feed-forward pass.

Method	DL3DV 16v				Mip-NeRF 360 32v			
	PSNR↑	LPIPS↓	AUC@3↑	Time (s)↓	PSNR↑	LPIPS↓	AUC@3↑	Time (s)↓
ZipSplat (full TokenBA)	24.79	0.175	0.954	1.80	24.43	0.210	0.737	4.76
w/o pose update	23.93	0.200	0.918	1.74	21.99	0.306	0.616	4.41
w/o alternating rounds	24.68	0.180	0.949	1.74	23.88	0.239	0.724	4.50
w/o joint refinement	23.16	0.211	0.944	0.27	23.16	0.254	0.685	0.66
Joint refinement w/o poses	24.69	0.178	0.944	1.80	24.15	0.218	0.685	4.70

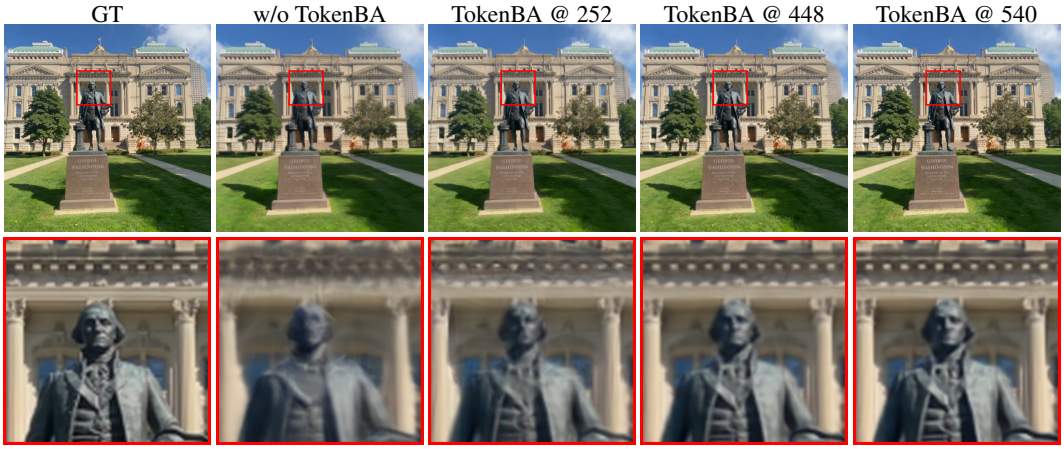


Figure 11: TokenBA at higher resolution on DL3DV with 64 input views. The encoder always runs at 252×252, while TokenBA fits the input views at the resolution of each column. Fitting at a higher resolution recovers the face of the statue from the same tokens.

on the predicted poses alone loses 0.9 dB in domain and 2.4 dB out of domain, where it even ends below TokenBA without any joint refinement. The joint refinement carries most of the remaining gain, 1.6 dB in domain and 1.3 dB out of domain, but also most of the time, since the alternating rounds alone finish in 0.27 s and 0.66 s. In domain, the predicted poses are already accurate, so the alternating rounds add only 0.1 dB. Out of domain, they recover the noisier initial poses and add 0.55 dB. Freezing the poses during the joint refinement costs 0.1 dB in domain and 0.3 dB out of domain, and the pose accuracy then stays at the value the alternating rounds reached.

C.4 TokenBA beyond the training resolution

The feed-forward encoder runs at its training resolution of 252×252, but the TokenBA rendering loss is resolution-agnostic. Refitting the tokens against higher-resolution input views recovers detail that the low-resolution pass cannot resolve. Figure 11 shows this on DL3DV at 64 views: at 252×252, TokenBA sharpens the prediction but leaves fine features such as the statue's face blurred, whereas refitting at 448×448 or 540×540 resolves them. The additional cost is modest – 4.6 times the pixels add only 1.6 times the refinement time (9.7 s to 15.6 s at 64 views). All quantitative results in this paper use TokenBA at 252×252.

C.5 Qualitative

Figure 12 shows target views of ZipSplat with and without TokenBA across view counts and out of domain. The feed-forward prediction already reconstructs the scene well, but it averages fine appearance that the model is uncertain about. TokenBA recovers this detail by fitting the input views through the frozen decoder, sharpening foliage and flowers, text, and reflective surfaces.

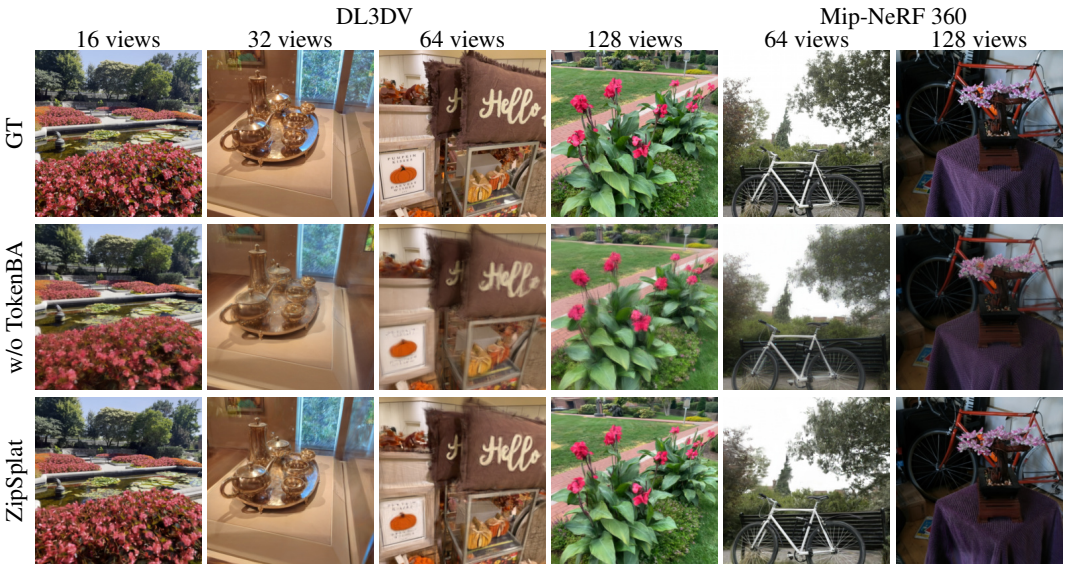


Figure 12: Effect of TokenBA. Target views of ZipSplat without and with TokenBA against the ground truth. The feed-forward prediction already recovers the scene well. TokenBA sharpens fine detail such as foliage and flowers, text, and reflective surfaces, both in domain and out of domain on Mip-NeRF 360.

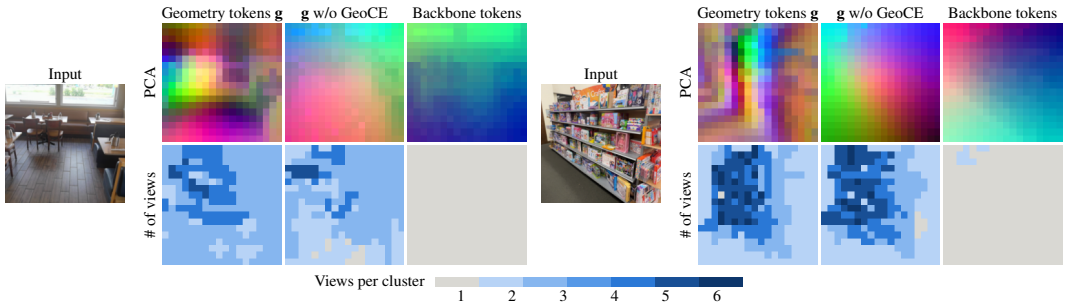


Figure 13: Token spaces and clusters. Top: per-patch PCA of the geometry tokens with and without GeoCE and of the pretrained backbone tokens. Without GeoCE, the tokens vary smoothly with depth. GeoCE separates the scene into distinct surface regions. Bottom: number of input views spanned by each patch's cluster at r=0.1. Clusters of geometry tokens merge patches across many views, whereas backbone token clusters stay within a single view.

D Clustering Analysis

This section looks at the tokens that ZipSplat clusters and at the clusters it builds from them. We use the two DA3-Base models of Table 4, trained with and without GeoCE, on DL3DV, and render without TokenBA.

D.1 GeoCE shapes the token space

GeoCE trains the geometry tokens so that patches showing the same 3D point have similar tokens, even when they come from different views. Figure 13 shows the effect. Without GeoCE, the tokens mostly encode depth and change smoothly across the image, and the backbone tokens behave the same way. With GeoCE, the tokens also tell apart different objects and surfaces at the same depth. To measure this, we take each patch and find the most similar token in the other views. With GeoCE, this token shows the same 3D point in 88% of cases, compared with 79% without GeoCE and 59% for the backbone tokens. Even without GeoCE, the geometry tokens find the same 3D point far more often than the backbone tokens, and their clusters still span several views. The rendering loss alone already shapes this space, since each scene token is the differentiable mean of the geometry tokens it merges (Section A.3), and the same tokens also serve as the attention keys. Merging tokens that do not show the same surface degrades the decoded Gaussians, so training pulls corresponding tokens

Table 11: Clustering method on DL3DV at 6 views. The same trained network is rendered from different groupings of its tokens. "Nearest" merges every token into its most similar seed without k-means iterations. "Seeds only" keeps the seed tokens and drops the rest.

Space	Method	Init.	Merge	PSNR↑ r=0.25	PSNR↑ r=0.1
Geometry tokens	k-means	Complexity	✓	22.92	21.51
Geometry tokens	k-means	Strided	✓	22.64	21.38
Backbone tokens	k-means	Strided	✓	22.10	20.67
Ground-truth 3D points	k-means	Strided	✓	22.50	21.26
Patch color	k-means	Strided	✓	19.37	17.66
Image position	Per-view grid	–	✓	21.84	20.34
Geometry tokens	Nearest	Strided	✓	22.57	21.20
Geometry tokens	Nearest	Farthest point	✓	22.59	21.17
Geometry tokens	Nearest	Random	✓	22.48	21.05
Geometry tokens	Seeds only	Strided		20.80	15.72
Geometry tokens	Seeds only	Random		19.91	15.10

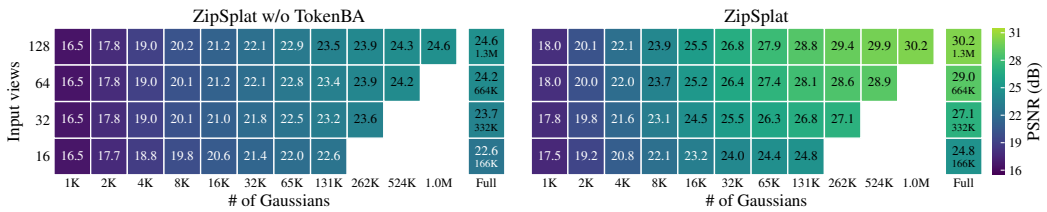


Figure 14: PSNR vs. input views and Gaussian budget on DL3DV, without (left) and with (right) TokenBA. For the same number of Gaussians, more input views give better renders.

together. GeoCE makes this correspondence explicit and sharper. Clustering inherits this, as patches merged into one cluster more often show the same part of the scene, 83% instead of 73% at r=0.1.

Compression should merge patches that show the same surface in different views, since they carry the same information. The bottom row of Fig. 13 shows how many views each cluster spans. Clusters of geometry tokens span several views, whereas clusters of backbone tokens rarely leave their own view. At r=0.1 and 6 views, 91% of the clusters span at least two views, as many as when clustering the ground-truth 3D points directly, compared with 38% for the backbone tokens. GeoCE matters more with more views: at 32 views, a cluster spans 6.4 views with GeoCE and 5.6 without, close to the 6.7 of the ground-truth 3D points.

D.2 Clustering method

ZipSplat compresses by running k-means on the geometry tokens and merging each cluster into one scene token. We test how much each part of this choice matters, namely the initialization, the space we cluster in, and the clustering itself. Table 11 keeps the network fixed and only changes how its tokens are grouped, and we compare each row with k-means on the geometry tokens under the same strided initialization. Complexity-weighted initialization adds 0.3 dB at r=0.25 and 0.1 dB at r=0.1. The space matters most. At r=0.1, clustering the backbone tokens costs 0.7 dB and the patch colors 3.7 dB, a per-view grid on image positions costs 1.0 dB, and even the ground-truth 3D points are 0.1 dB worse than the geometry tokens. The clustering algorithm matters little, since merging every token into its nearest seed without k-means iterations costs only 0.1 to 0.2 dB. Merging itself is essential, as keeping only the seed tokens loses 1.8 dB at r=0.25 and 5.7 dB at r=0.1.

D.3 Compression results

Figure 14 shows the PSNR of ZipSplat for every combination of input views and Gaussian budget. For the same number of Gaussians, more input views give better renders. At 131K Gaussians, going from 16 to 128 views adds 0.9 dB without TokenBA and 4.0 dB with it. TokenBA helps more as the number of views and Gaussians grows, from 1.0 dB at 16 views and 1K Gaussians to 5.7 dB at 128 views and full capacity. Without TokenBA, only budgets of 2K Gaussians or fewer do not improve with more views. Figure 15 shows two views at full capacity and at 4× and 20× compression. What a scene loses depends on its content: large flat regions survive 20× compression almost unchanged, whereas grass and thin structures need over 8K Gaussians, and TokenBA cannot fully recover them.

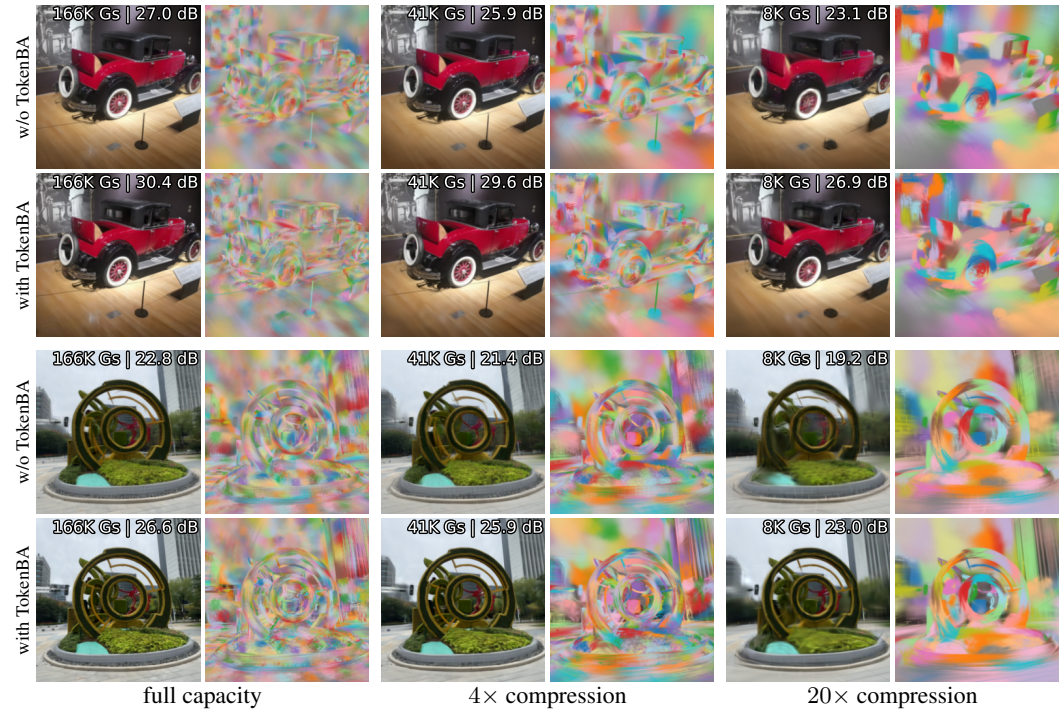


Figure 15: Compression on DL3DV at 16 views. Two target views at full capacity, $4\times$ compression (r=0.25), and $20\times$ compression (r=0.05), each as render (left) and token map (right), without and with TokenBA. Scenes respond differently to compression: the car scene has large flat regions, so $20\times$ compression changes the render little, whereas the grass and thin structures of the second scene need more than 8K Gaussians, and TokenBA cannot fully bring them back.

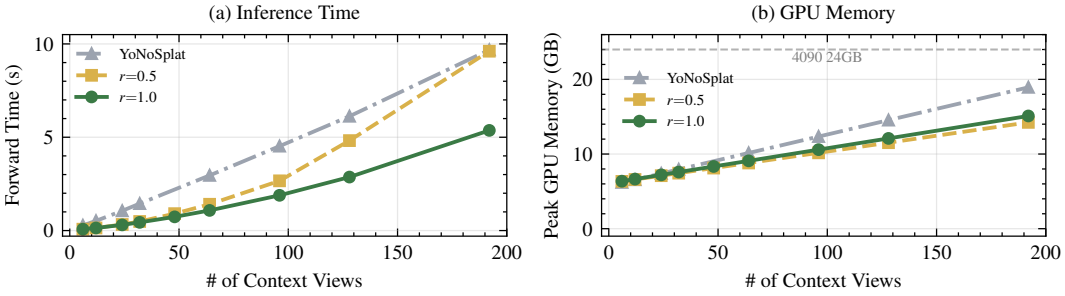


Figure 16: Forward time and memory vs. input views on DL3DV. (a) Forward time and (b) peak GPU memory of ZipSplat on an RTX 4090 at full capacity and at r=0.5, with YoNoSplat (Ye et al., 2026) for reference. Full capacity skips clustering, whereas r=0.5 pays for its k-means iterations at high view counts.

E Inference Time and Rendering Speed

Setup. All measurements run on a single NVIDIA RTX 4090 (24 GB) at 252×252 resolution with batch size 1, with the backbone and fusion in bf16. Each configuration averages 15 timed forward passes after 5 warm-up passes on a single DL3DV scene. The forward pass covers the prediction of the Gaussians and excludes data loading, TokenBA and the evaluation renders. We time full capacity (r=1.0), which keeps every token and skips clustering, and a fixed ratio of r=0.5.

Inference cost. Forward time grows with the number of input views (Fig. 16a), from 0.07 s at 6 views to 5.4 s at 192 views at full capacity. Full capacity is as fast as the compressed ratios because it skips clustering, and the rest of the network is shallow. The 40-layer backbone takes about three quarters of the time, and the three fusion blocks, the color attention and the Gaussian head add the remaining quarter even though every token is a query. Compression can only shrink this last quarter, and it first pays for k-means. At r=0.5 this costs little up to 32 views, but the k-means iterations grow with the number of tokens times the number of clusters. At 192 views, r=0.5 takes 9.6 s, the same as YoNoSplat, and the 4.2 s it adds over full capacity are the k-means iterations, which compare all

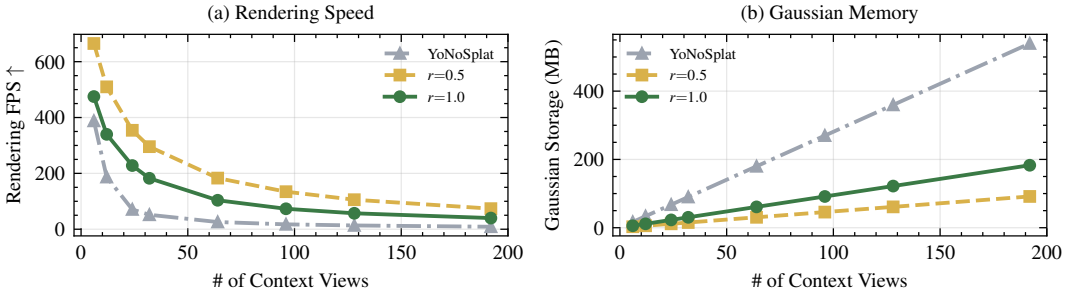


Figure 17: Rendering speed and storage vs. input views on DL3DV. (a) Rendering FPS and (b) storage of the predicted Gaussians at full capacity and at r=0.5, with YoNoSplat (Ye et al., 2026) for reference. The Gaussian count grows linearly with the views, and r=0.5 halves the storage and nearly doubles the rendering speed.

Table 12: Full DL3DV results. All methods at 16, 32, 64, and 128 input views. Posed methods receive ground-truth cameras. #Gs is the total number of predicted Gaussians.

Method	Posed	16v				32v				64v				128v			
		#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓
DepthSplat (Xu et al., 2025)	✓	1.8M	19.64	0.660	0.278	3.7M	18.61	0.623	0.318	7.3M	17.32	0.555	0.390	14.7M	15.98	0.461	0.489
ZPressor (Wang et al., 2025b)	✓	688K	18.44	0.575	0.339	688K	18.34	0.569	0.344	688K	18.36	0.568	0.345	688K	18.44	0.570	0.344
VolSplat (Wang et al., 2026)	✓	1.0M	15.42	0.385	0.565	2.1M	15.64	0.399	0.560	4.2M	15.55	0.398	0.566	OOM			
GlobalSplat (Itkin et al., 2026)	✓	16K	15.48	0.397	0.630	16K	15.68	0.402	0.625	16K	15.72	0.402	0.626	16K	15.80	0.404	0.628
TokenGS w/o TTO (Ren et al., 2026)	✓	262K	20.16	0.614	0.433	262K	20.46	0.626	0.428	262K	20.56	0.631	0.427	262K	20.62	0.634	0.425
TokenGS (Ren et al., 2026)	✓	262K	21.41	0.695	0.345	262K	22.28	0.727	0.332	262K	22.66	0.740	0.328	262K	22.86	0.748	0.324
AnySplat (Jiang et al., 2025)		2.3M	18.40	0.559	0.314	4.1M	19.60	0.603	0.283	7.2M	19.91	0.617	0.278	12.3M	19.95	0.622	0.279
EcoSplat (r=0.4) (Park et al., 2026)		419K	14.05	0.319	0.605	839K	13.69	0.302	0.626	1.7M	13.06	0.275	0.647	3.4M	12.84	0.268	0.661
EcoSplat (r=0.95) (Park et al., 2026)		971K	13.84	0.305	0.611	1.9M	13.44	0.288	0.632	3.9M	12.78	0.262	0.651	7.8M	12.61	0.258	0.662
C3G (An et al., 2026)		2K	12.71	0.286	0.649	2K	11.97	0.258	0.661	2K	11.56	0.243	0.672	2K	11.51	0.244	0.678
DA3 (Lin et al., 2026)		4.1M	20.34	0.652	0.276	8.1M	20.82	0.673	0.255	16.3M	20.59	0.666	0.257	32.5M	20.09	0.646	0.271
YoNoSplat (Ye et al., 2026)		803K	20.36	0.638	0.299	1.6M	20.89	0.664	0.278	3.2M	21.00	0.674	0.275	6.4M	20.42	0.661	0.293
ZipSplat w/o TokenBA		166K	22.61	0.725	0.213	332K	23.65	0.763	0.182	664K	24.25	0.786	0.165	1.3M	24.56	0.800	0.155
ZipSplat (r=0.1)		17K	23.22	0.739	0.293	33K	25.51	0.809	0.229	66K	27.39	0.857	0.179	133K	28.76	0.887	0.145
ZipSplat (r=0.5)		83K	24.57	0.798	0.200	166K	26.89	0.856	0.153	332K	28.74	0.893	0.122	664K	29.98	0.914	0.102
ZipSplat		166K	24.79	0.808	0.175	332K	27.11	0.865	0.133	664K	28.99	0.900	0.106	1.3M	30.22	0.920	0.090
ZipSplat	✓	166K	**24.88**	**0.813**	**0.173**	332K	**27.23**	**0.870**	**0.130**	664K	**29.14**	**0.905**	**0.104**	1.3M	**30.41**	**0.926**	**0.088**

62K tokens with 31K cluster centers in every iteration (Section C.2). Peak GPU memory (Fig. 16b) grows from 6.4 GB at 6 views to 15.1 GB at 192 views at full capacity, and r=0.5 needs slightly less. YoNoSplat reaches 18.9 GB at 192 views.

Output representation. The Gaussian count sets the cost of the final representation (Fig. 17). At full capacity it grows linearly with the number of input views, from 62K Gaussians at 6 views to 2.0M at 192 views (183 MB, 40 FPS at 252×252). Halving the ratio halves the storage and nearly doubles the rendering speed, to 92 MB and 74 FPS at 192 views. At 24 views, full capacity renders 249K Gaussians at 228 FPS and r=0.5 renders 124K at 354 FPS, well within the limits of real-time viewers. YoNoSplat produces 9.6M Gaussians at 192 views (539 MB) and renders at 8.9 FPS, about 5× more primitives and 4.5× slower than ZipSplat at full capacity.

F More Results

The tables in this section extend the main results to every baseline, every view count, and every variant of ZipSplat. Table 12 covers the DL3DV ladder at 16, 32, 64, and 128 views, and Tables 13 to 15 the remaining benchmarks. Each table lists ZipSplat without TokenBA, at fixed ratios r=0.5 and 0.1 or at the fixed budget of 262K Gaussians of TokenGS (with TokenBA), and with ground-truth cameras (posed).

F.1 DL3DV

No baseline exceeds 23 dB at any view count, and no pose-free baseline exceeds 21.0 dB. DepthSplat, DA3, and YoNoSplat decline beyond 32 or 64 views, and the fixed-budget methods gain at most 1.45 dB (TokenGS) across the ladder, whereas ZipSplat gains 1.95 dB from 16 to 128 views without TokenBA and 5.43 dB with it. A fixed ratio of r=0.5 stays within 0.25 dB of full capacity with half the Gaussians, and r=0.1 stays within 1.6 dB with a tenth while remaining at least 1.8 dB above every baseline. Ground-truth cameras add at most 0.19 dB, so the predicted cameras cost almost nothing in domain.

Table 13: Full RealEstate10K results at 6 input views. Posed methods receive ground-truth cameras; NoPoSplat receives only intrinsics. #Gs is the total number of predicted Gaussians.

Method	Posed	RE10K 6v			
		#Gs	PSNR↑	SSIM↑	LPIPS↓
DepthSplat (Xu et al., 2025)	✓	688K	24.62	0.873	0.138
ZPressor (Wang et al., 2025b)	✓	688K	24.89	0.864	0.141
VolSplat (Wang et al., 2026)	✓	393K	22.98	0.807	0.230
GlobalSplat (Itkin et al., 2026)	✓	16K	23.68	0.782	0.232
TokenGS w/o TTO (Ren et al., 2026)	✓	262K	26.57	0.865	0.199
TokenGS (Ren et al., 2026)	✓	262K	27.59	0.899	0.145
NoPoSplat (Ye et al., 2025a)		393K	20.22	0.680	0.311
AnySplat (Jiang et al., 2025)		775K	22.60	0.791	0.189
EcoSplat (r=0.4) (Park et al., 2026)		157K	21.73	0.728	0.267
EcoSplat (r=0.95) (Park et al., 2026)		362K	21.82	0.733	0.265
C3G (An et al., 2026)		2K	20.64	0.663	0.318
DA3 (Lin et al., 2026)		1.5M	21.37	0.726	0.229
YoNoSplat (Ye et al., 2026)		301K	25.05	0.830	0.154
ZipSplat w/o TokenBA		62K	26.86	0.856	0.134
ZipSplat (r=0.1)		6K	26.76	0.846	0.198
ZipSplat (r=0.5)		31K	28.34	0.880	0.130
ZipSplat		62K	28.50	0.882	0.120
ZipSplat	✓	62K	**30.13**	**0.924**	**0.102**

Table 14: Full Mip-NeRF 360 results at 32, 64, and 128 input views. No method is trained on this dataset. Posed methods receive ground-truth cameras. #Gs is the total number of predicted Gaussians. The per-scene 3DGS fit is a grey reference excluded from the ranking.

Method	Posed	32v				64v				128v			
		#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓
$3DGS_{30k}$ (Kerbl et al., 2023)	✓	1.0M	21.28	0.678	0.282	1.2M	24.84	0.806	0.188	1.3M	27.21	0.862	0.137
DepthSplat (Xu et al., 2025)	✓	3.7M	16.33	0.443	0.455	7.3M	15.62	0.405	0.499	14.7M	14.65	0.356	0.585
ZPressor (Wang et al., 2025b)	✓	688K	17.29	0.445	0.446	688K	17.34	0.447	0.448	688K	16.93	0.418	0.464
GlobalSplat (Itkin et al., 2026)	✓	16K	14.91	0.375	0.753	16K	14.56	0.360	0.748	16K	15.14	0.380	0.747
TokenGS w/o TTO (Ren et al., 2026)	✓	262K	19.77	0.507	0.569	262K	19.76	0.510	0.564	262K	19.92	0.517	0.561
TokenGS (Ren et al., 2026)	✓	262K	22.84	0.648	0.434	262K	23.07	0.656	0.432	262K	23.22	0.663	0.424
AnySplat (Jiang et al., 2025)		4.5M	19.10	0.537	0.322	8.2M	19.91	0.551	0.302	14.5M	20.03	0.556	0.302
EcoSplat (Park et al., 2026)		839K	14.98	0.320	0.650	1.7M	14.85	0.308	0.657	3.4M	14.36	0.297	0.681
C3G (An et al., 2026)		2K	14.59	0.315	0.690	2K	14.48	0.318	0.693	2K	14.28	0.316	0.702
DA3 (Lin et al., 2026)		8.1M	21.48	0.603	0.266	16.3M	21.03	0.585	0.277	32.5M	20.62	0.579	0.287
YoNoSplat (Ye et al., 2026)		1.6M	17.61	0.411	0.484	3.2M	17.73	0.417	0.479	6.4M	17.54	0.416	0.494
ZipSplat w/o TokenBA		332K	22.30	0.606	0.256	664K	22.53	0.620	0.244	1.3M	22.81	0.634	0.231
ZipSplat (262K)		262K	24.37	0.695	0.216	262K	25.16	0.719	0.217	262K	25.66	0.733	0.216
ZipSplat		332K	24.43	0.696	0.210	664K	25.24	0.723	0.185	1.3M	25.81	0.739	0.169
ZipSplat	✓	332K	**26.57**	**0.820**	**0.148**	664K	**27.38**	**0.845**	**0.125**	1.3M	**28.35**	**0.870**	**0.106**

F.2 RealEstate10K

Without TokenBA, ZipSplat already exceeds TokenGS w/o TTO and YoNoSplat with 62K Gaussians, four and five times fewer. At r=0.1, 6K Gaussians reach the PSNR of the full feed-forward model. Ground-truth cameras add 1.63 dB here against 0.1 dB on DL3DV, in line with the lower pose accuracy at six views (Table 16).

F.3 Mip-NeRF360

Without TokenBA, ZipSplat exceeds every feed-forward baseline at every view count, and its PSNR rises from 32 to 128 views while DA3, which shares our backbone, declines by 0.86 dB. At the fixed budget of 262K Gaussians, matching TokenGS, ZipSplat stays within 0.15 dB of full capacity at every view count, with five times fewer Gaussians at 128 views. With ground-truth cameras, ZipSplat gains 2.1 to 2.5 dB and exceeds the per-scene 3DGS fit at every view count, so out of domain the gap to oracle cameras is ten times larger than on DL3DV.

F.4 ScanNet++

ZipSplat without cameras exceeds the per-scene 3DGS fit at every view count, by 2.25 to 3.26 dB. Without TokenBA it also exceeds AnySplat and DA3, although both train on ScanNet++. Ground-truth cameras add 2.99 and 2.42 dB at 32 and 64 views.

Table 15: Full ScanNet++ results on iPhone captures at 32, 64, and 128 input views. Posed methods receive ground-truth cameras. #Gs is the total number of predicted Gaussians. The per-scene 3DGS fit is a grey reference excluded from the ranking. *: AnySplat and DA3 are trained on ScanNet++.

Method	Posed	32v				64v				128v			
		#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓	#Gs	PSNR↑	SSIM↑	LPIPS↓
$3DGS_{30k}$ (Kerbl et al., 2023)	✓	1.2M	20.11	0.690	0.365	1.6M	21.16	0.732	0.345	1.7M	22.14	0.775	0.324
DepthSplat (Xu et al., 2025)	✓	3.7M	17.96	0.705	0.393	7.3M	17.35	0.688	0.422	14.7M	16.65	0.660	0.469
ZPressor (Wang et al., 2025b)	✓	688K	15.42	0.575	0.468	688K	16.56	0.627	0.432	688K	16.54	0.628	0.432
GlobalSplat (Itkin et al., 2026)	✓	16K	14.80	0.606	0.633	16K	14.93	0.612	0.632	16K	14.88	0.611	0.636
TokenGS w/o TTO (Ren et al., 2026)	✓	262K	19.80	0.714	0.505	262K	19.64	0.711	0.513	262K	19.59	0.712	0.515
TokenGS (Ren et al., 2026)	✓	262K	22.14	0.772	0.422	262K	22.49	0.782	0.426	262K	22.71	0.786	0.427
AnySplat* (Jiang et al., 2025)		3.5M	16.63	0.669	0.391	6.9M	18.36	0.694	0.348	12.8M	18.53	0.692	0.354
EcoSplat (Park et al., 2026)		839K	14.07	0.515	0.642	1.7M	13.67	0.498	0.654	3.4M	13.75	0.506	0.667
C3G (An et al., 2026)		2K	12.24	0.487	0.627	2K	12.00	0.479	0.635	2K	11.71	0.468	0.641
DA3* (Lin et al., 2026)		8.1M	18.48	0.683	0.394	16.3M	19.15	0.702	0.362	32.5M	19.12	0.703	0.361
YoNoSplat (Ye et al., 2026)		1.6M	16.13	0.622	0.516	3.2M	16.29	0.626	0.506	6.4M	16.12	0.623	0.513
ZipSplat w/o TokenBA		332K	20.34	0.729	0.324	664K	21.42	0.758	0.278	1.3M	21.76	0.770	0.265
ZipSplat (262K)		262K	22.46	0.780	0.299	262K	24.29	0.821	0.263	262K	25.23	0.842	0.260
ZipSplat		332K	22.36	0.779	0.298	664K	24.36	0.826	0.249	1.3M	25.40	0.848	0.235
ZipSplat	✓	332K	**25.35**	**0.863**	**0.209**	664K	**26.78**	**0.891**	**0.186**	1.3M	**27.93**	**0.905**	**0.176**

Table 16: Full pose accuracy. AUC of the relative rotation and translation-direction error over all input-view pairs, pooled over all scenes, at thresholds from 1° to 30°. DL3DV uses 64 input views; RealEstate10K uses 6.

Method	DL3DV 64v							RE10K 6v						
AUC↑	1°	3°	5°	10°	15°	20°	30°	1°	3°	5°	10°	15°	20°	30°
AnySplat (Jiang et al., 2025)	0.071	0.357	0.550	0.755	0.833	0.873	0.914	0.130	0.378	0.530	0.715	0.795	0.839	0.886
DA3 (Lin et al., 2026)	0.930	0.969	0.980	0.989	**0.993**	**0.994**	**0.996**	0.546	0.741	0.817	0.890	0.920	0.935	0.953
YoNoSplat (Ye et al., 2026)	0.466	0.754	0.842	0.915	0.941	0.955	0.969	0.361	0.673	0.779	0.876	0.913	0.933	0.953
ZipSplat w/o TokenBA	0.827	0.920	0.948	0.972	0.980	0.985	0.989	0.706	0.843	0.890	0.936	0.954	0.964	0.975
ZipSplat	**0.950**	**0.976**	**0.983**	**0.990**	**0.993**	**0.994**	**0.996**	**0.738**	**0.855**	**0.897**	**0.939**	**0.956**	**0.965**	**0.976**

F.5 POSE ACCURACY

With TokenBA, ZipSplat is the most accurate at every threshold up to 10° on both datasets and ties DA3 from 15° on DL3DV. Our feed-forward camera head is initialized from DA3's. At 64 views on DL3DV it trails DA3 at 1° (0.827 against 0.930), while at six views on RealEstate10K it is far ahead (0.706 against 0.546). TokenBA lifts it above DA3 in both cases.

G Additional Qualitative Results

Figure 18 extends Fig. 5 with more DL3DV scenes at 16, 32, 64, and 128 input views. We show additional qualitative comparisons on RealEstate10K in Fig. 19.

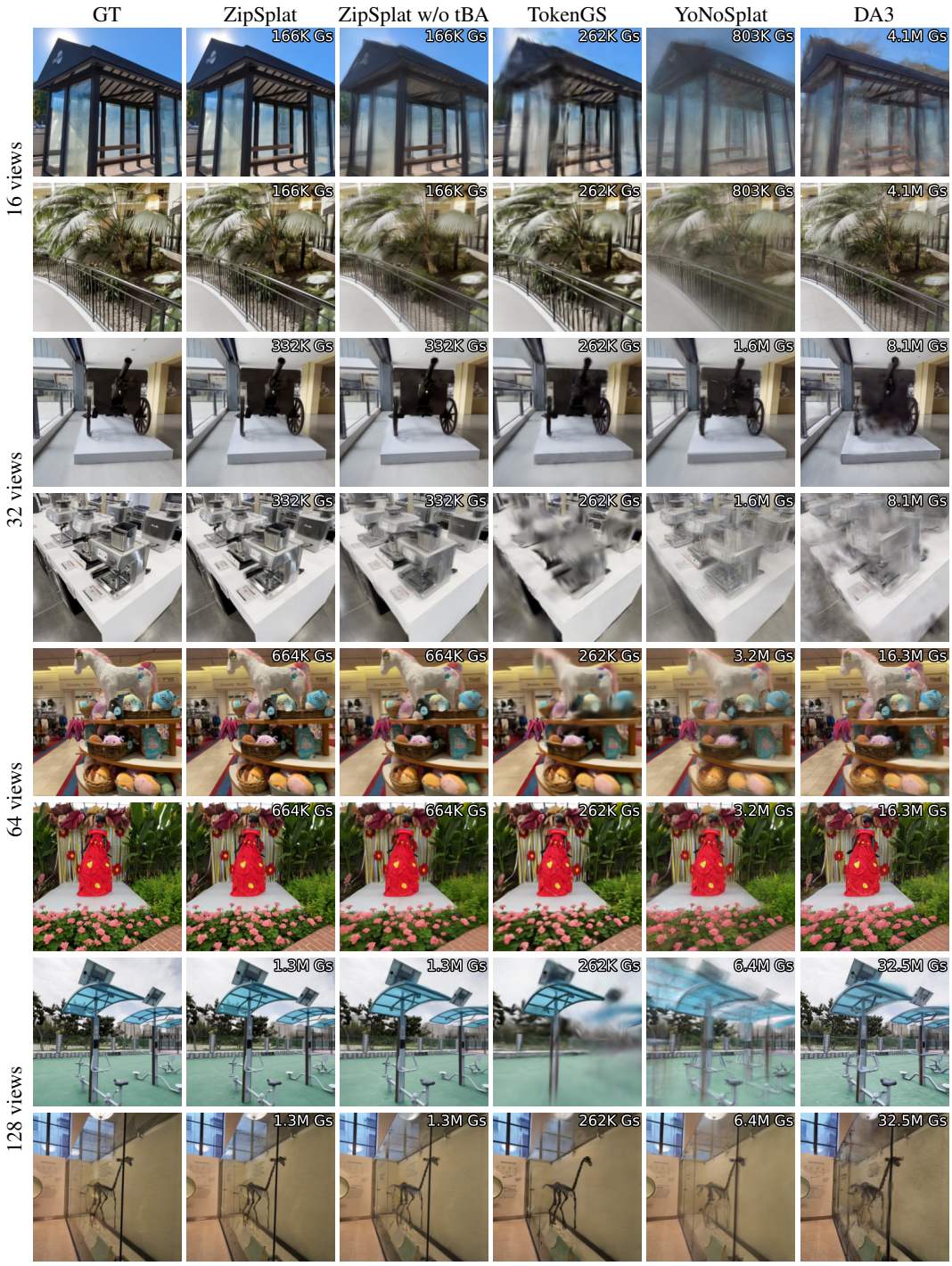


Figure 18: Additional qualitative comparisons on DL3DV (16 to 128 input views).

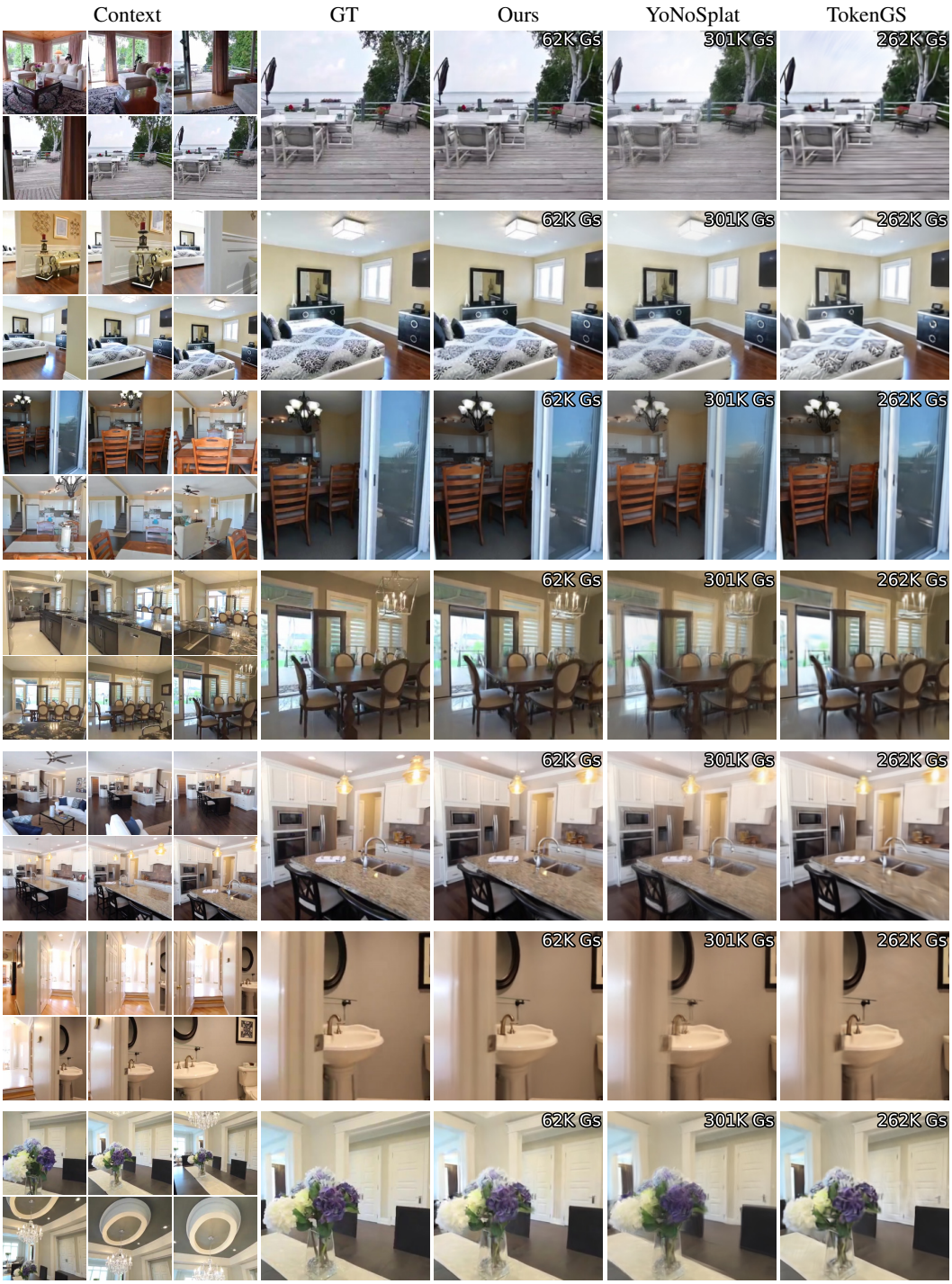


Figure 19: Additional qualitative comparisons on RealEstate10K (6 input views).