FINESPLAT: FINE PATCHES, LOCAL GROUPS, DETAILED SPLATS

Anonymous authors
Paper under double-blind review

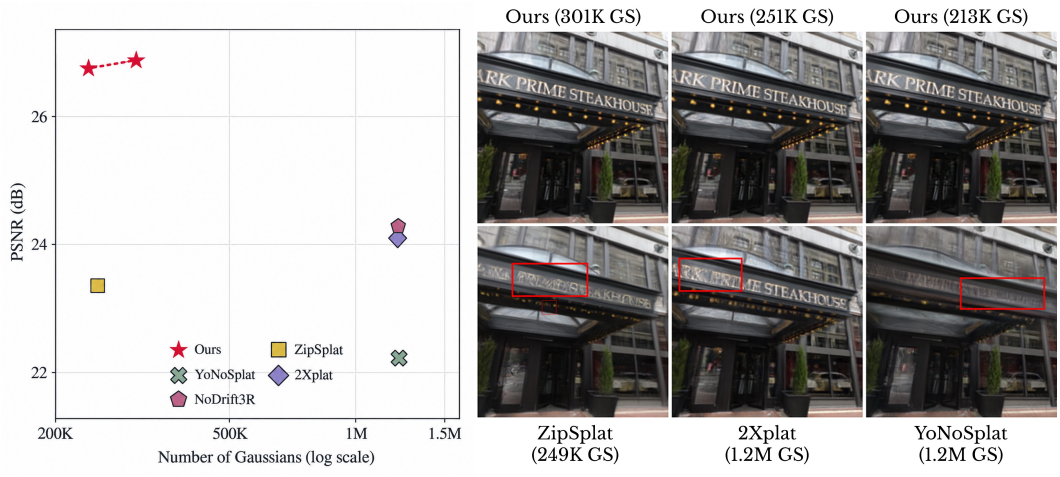

Figure 1: Left: PSNR vs. number of Gaussians on DL3DV with 24 input views in the pose-free setting (224×224 resolution). Right: Qualitative novel-view rendering comparison, highlighting how *FineSplat* better preserves fine details with fewer Gaussians.

ABSTRACT

Feed-forward 3D Gaussian Splatting (3DGS) promises instant scene reconstruction with high-quality novel-view synthesis. Yet, fine-grained geometry and appearance remain difficult to recover, motivating us to revisit how feed-forward models represent, aggregate, and refine detailed visual information. We introduce *FineSplat*, a feed-forward 3DGS framework built around fine-grained image patches. Since global multi-view attention becomes prohibitively expensive at such high token resolutions, we develop a *dual-grouping* strategy that combines two complementary forms of locality: view-wise grouping enables efficient interaction among neighboring views, while 3D spatial grouping reconnects geometrically related tokens that may never have interacted under view-wise grouping by lifting them into the 3D scene space. We further leverage reconstruction error as direct feedback, providing an explicit rendering-based signal to recover fine-grained structures and appearance, thereby overcoming the limitations of one-pass feed-forward prediction. Finally, a compact Gaussian generation scheme preserves the benefits of fine-grained features without unnecessarily increasing the number of primitives. Together, these designs enable *FineSplat* to recover richer geometry and appearance while remaining efficient and compact, achieving state-of-the-art results by a substantial margin, under the same or smaller Gaussian budgets than existing methods.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023; Yu et al., 2024) has emerged as a powerful representation for high-quality and real-time novel view synthesis, combining an explicit 3D scene representation with efficient differentiable rasterization (Ye et al., 2025b). Since conventional 3DGS

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.

relies on iterative per-scene optimization to achieve high reconstruction quality, it typically requires substantial computational resources for every new scene, limiting its scalability and applicability to various practical applications. To overcome these limitations, recent works have explored feed-forward 3DGS, where Gaussian representations are directly predicted from a set of input images in a single network forward pass. These approaches have rapidly advanced in both posed (Charatan et al., 2024; Chen et al., 2024a; Ziwen et al., 2025; Kang et al., 2026a;b) and unposed (Zhang et al., 2025; Ye et al., 2025a; 2026; An et al., 2026) settings, enabling fast reconstruction with promising generalization. However, replacing iterative per-scene refinement with a single feed-forward prediction places a much greater burden on the network, requiring it to resolve complex multi-view ambiguities and recover fine-grained scene details all at once. As a result, existing feed-forward approaches still exhibit a considerable reconstruction-quality gap compared with per-scene optimization methods, particularly in recovering fine-grained geometry and appearance details.

Many recent feed-forward 3DGS methods build upon powerful 3D foundation models, benefiting from their strong geometric priors and multi-view reasoning capabilities. However, these models commonly inherit the relatively coarse patch resolution of their geometry backbones, typically using 14×14 or 16×16 image patches (Wang et al., 2025a; Lin et al., 2026; Wang et al., 2026c). While such coarse tokenization is effective for capturing large-scale scene geometry, each token must compress a spatially broad image region that may contain diverse local structures and appearance variations. This compression can discard high-frequency cues that are particularly important for reconstructing thin structures, sharp boundaries, textures, and other fine-grained details. Finer patch tokens may therefore offer a promising avenue for narrowing this gap by preserving richer local visual cues.

Yet, finer tokenization introduces a fundamental computational challenge. Most multi-view transformers rely on global self-attention across tokens from multiple input images, whose quadratic complexity quickly becomes prohibitive as the number of views, image resolution, and token density increase. A practical alternative is to restrict the attention scope. For example, MVP (Kang et al., 2026b) partitions input views into small groups and performs fine-grained multi-view attention only within each group, substantially improving the efficiency of high-resolution multi-view reconstruction while maintaining strong accuracy. Such view-wise grouping is particularly effective when the inputs are sequentially captured and nearby views are likely to observe overlapping scene regions. However, the restriction is ultimately imposed in view space rather than scene space: tokens corresponding to the same 3D surface may fall into different view groups and therefore never directly interact. Consequently, any fixed view grouping inevitably creates blind spots, excluding potentially relevant correspondences simply because they lie across group boundaries.

In this work, we propose *FineSplat*, which addresses this challenge through a *dual-grouping* strategy that exploits two complementary forms of locality. First, we perform fine-grained multi-view interaction within small groups of neighboring views, enabling efficient attention over high-resolution tokens. We then lift the resulting tokens into 3D space using their predicted scene locations and regroup them according to 3D spatial proximity. This allows tokens from different, potentially distant view groups to interact when they correspond to nearby regions of the underlying 3D scene, recovering geometric relationships missed by view-wise grouping alone. By combining *view-wise locality* with *3D spatial locality*, *FineSplat* enables rich interactions among fine-grained tokens without the prohibitive cost of global multi-view attention.

3D reconstruction provides a natural signal for assessing the quality of a prediction: a faithful reconstruction should reproduce the observed input images when rendered from their corresponding viewpoints. Conventional per-scene optimization repeatedly exploits this signal, comparing rendered and observed images to identify residual errors and progressively correct the scene representation. A one-pass feed-forward model, in contrast, typically produces its reconstruction without an opportunity to inspect and correct its own errors. Motivated by this observation and recent work (Xu et al., 2026), we therefore explicitly leverage this readily available feedback as a direct signal in the second stage of our proposed framework. This provides the network with a second chance to recover structures and fine details that were missed in the first stage, without resorting to costly iterative optimization.

We further introduce a compact Gaussian generation scheme that predicts only a small number of Gaussians from each patch token, rather than densely generating pixel-aligned Gaussians. This design allows the model to retain the benefits of fine-grained image features without increasing

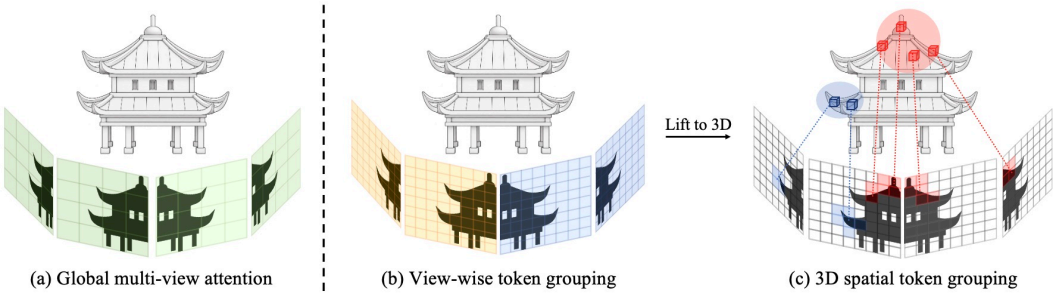

Figure 2: **Dual grouping strategy.** (a) Global multi-view attention allows unrestricted interaction among tokens from all views without explicitly accounting for geometric correspondence, but is typically limited to coarse tokens due to its computational cost. (b) View-wise grouping processes tokens independently within each originating view, preserving view-specific structure while remaining computationally feasible for fine-grained tokens. (c) 3D spatial grouping lifts tokens into 3D using the estimated camera geometry and groups spatially neighboring tokens across views, enabling geometry-aware cross-view interactions at fine granularity.

the number of Gaussian primitives. Together, the fine-grained representation, dual-grouping strategy, reconstruction error-driven correction, and compact Gaussian generation enable substantially improved reconstruction quality while maintaining an efficient and compact 3D representation. Extensive experiments demonstrate that our method achieves state-of-the-art rendering quality by a substantial margin in both posed and unposed settings, while using the same or fewer Gaussians than competing methods.

2 METHOD

2.1 STAGE 1: VIEW-WISE TOKEN GROUPING

Global multi-view attention becomes prohibitively expensive with fine-grained tokens. In the first stage, we restrict the attention scope and adopt a view-wise token grouping strategy. Specifically, we leverage the multi-view pyramid architecture (MVP) (Kang et al., 2026b). It performs fine-grained attention within small groups of views, while allowing information exchange across all views through coarser global tokens. This balances local detail preservation with global multi-view reasoning. It organizes its encoder as a three-level pyramid, and we denote the feature map of level $\ell \in \{1, 2, 3\}$ as F^ℓ, from the finest (F^1) to the coarsest (F^3). In more details, given N input images $I \in \mathbb{R}^{N\times H\times W\times 3}$ and patch size p, the first stage fuses the three levels coarse-to-fine features into $M = \frac{NHW}{p^2}$ fine-grained tokens $z^{(0)} \in \mathbb{R}^{M\times d}$,

$$z^{(0)} = \mathrm{fuse}\big(\mathrm{up}(\mathrm{fuse}(\mathrm{up}(F^3) + F^2)) + F^1\big), \tag{1}$$

where $\mathrm{up}(\cdot)$ denotes bilinear upsampling and $\mathrm{fuse}(\cdot)$ a convolutional fusion block, which are used to match the token resolutions.

The fixed view grouping strategy inevitably leaves blind spots: fine-grained tokens in different groups cannot directly interact, potentially missing relevant observations from other input views. Thus, view-wise grouping alone cannot fully realize fine-grained multi-view reasoning over all input views. To recover these missing interactions, we pass the fine-grained tokens to the subsequent 3D spatial grouping stage.

2.2 STAGE 2: 3D SPATIAL TOKEN GROUPING

Starting from the fine-grained tokens $z^{(0)}$ produced by the first stage, we first decode an initial set of 3D Gaussians. Specifically, a lightweight Gaussian head $\mathcal{D}$, implemented as a linear decoder, maps each token to K Gaussian primitives and their attributes. We then use the predicted Gaussian centers to lift the corresponding tokens into 3D space, where tokens are regrouped according to their spatial proximity in 3D world coordinates. We then perform self-attention within each 3D group,

enabling fine-grained interactions between geometrically related tokens even when they originate from different view groups. To further refine the current estimate, we augment this interaction with reconstruction-error information obtained by rendering the input views. The updated tokens are subsequently decoded into a new set of Gaussian primitives, and this process of Gaussian prediction, 3D grouping, error-guided interaction, and token update is repeated for multiple iterations.

Lifting tokens to 3D. To reason about the tokens in a common 3D space, we anchor each token at the mean position of the Gaussians it produces. Given j-th token at i-th iteration, $z_j^{(i)} \in \mathbb{R}^d$,

$$\{\mu_{jk}^{(i)}\}_{k=1}^{K} = \mathcal{D}(z_j^{(i)}), \quad p_j^{(i)} = \frac{1}{K}\sum_{k=1}^{K} \mu_{jk}^{(i)}, \tag{2}$$

where $\mu_{jk}^{(i)} \in \mathbb{R}^3$ is the position of each predicted Gaussian. Each token is then projected by a linear layer, yielding a point cloud $\{(p_j^{(i)}, z_j^{(i)})\}_{j=1}^{M}$ whose points carry the fine-grained tokens as features.

Error feature maps. To measure how well the current Gaussians explain the observations, we render them into every input view. For the n-th context view, we define a signed error map between the input image I_n and the rendered image $\hat{I}_n$, computed as $E_n = \hat{I}_n - I_n$. To complement this pixel-level discrepancy with structural cues, we additionally compute a feature-level difference $\Delta\phi_n = \phi(\hat{I}_n) - \phi(I_n)$, where $\phi(\cdot)$ is a pretrained ResNet-18 (He et al., 2016) encoder. These two terms are combined with the finest-scale features from the first stage, F_n^1, which provide the image context in which the errors occur. We obtain the error feature map as follows,

$$F_n^{\mathrm{err}} = \left[\, \mathcal{E}_{\mathrm{err}}(E_n, \Delta\phi_n)\,;\ \mathcal{E}_{\mathrm{feat}}(F_n^1) \,\right] \in \mathbb{R}^{\frac{H}{p}\times\frac{W}{p}\times d_e}, \tag{3}$$

where F_n^1 denotes the finest-scale features from the first stage for an input view n, $\mathcal{E}_{\mathrm{err}}$ and $\mathcal{E}_{\mathrm{feat}}$ are linear embedding layers, and $[\cdot;\cdot]$ represents the feature concatenation.

Gathering errors across views. A Gaussian affects the rendering of every view in which it is visible, not only the view from which it was predicted. We therefore gather error features for each token from both its own view and the other views that observe it. Let $\pi_n(\cdot)$ denote the projection onto the error feature map of view n, and let $v_{jn} \in \{0, 1\}$ indicate whether $p_j^{(i)}$ lies inside the view n's frustum and passes an occlusion test against the rendered depth. The aggregated error feature is obtained by averaging the bilinearly sampled features over the visible views,

$$e_j^{\mathrm{agg}} = \frac{\sum_{n=1}^{N} v_{jn}\, F_n^{\mathrm{err}}\big(\pi_n(p_j^{(i)})\big)}{\sum_{n=1}^{N} v_{jn}}, \tag{4}$$

where $F_n^{\mathrm{err}}(\cdot)$ reads a feature given a projected coordinate. In addition, we take the error feature at the patch from which $z_j^{(i)}$ originates, denoted as e_j^{self}. While e_j^{self} provides a direct signal from the source view, e_j^{agg} summarizes how consistently the token explains the other views. The two are fused into the final error feature $e_j = \mathrm{Linear}\big([e_j^{\mathrm{self}}; e_j^{\mathrm{agg}}]\big)$.

3D spatial token grouping and update. Each lifted point is assigned the feature $h_j = \mathrm{Linear}\big([z_j^{(i)}; e_j]\big)$. Neighboring tokens in 3D often originate from different views and describe the same surface, so correcting them jointly is more effective than correcting each token independently. We thus exchange information among nearby points using serialized attention (Wu et al., 2024), which orders the points along a space-filling curve, partitions them into groups of S consecutive points, and applies self-attention within each group. Compared with explicit k-nearest-neighbor search, this provides local 3D context at a much lower cost and scales favorably with the number of tokens. The updated features are projected to a token residual by a linear layer,

$$\Delta z_j^{(i)} = \mathrm{Linear}\big(\mathrm{SerialAttn}(h_j)\big), \qquad z_j^{(i+1)} = z_j^{(i)} + \Delta z_j^{(i)}. \tag{5}$$

Then, the updated tokens are decoded into Gaussians with the shared Gaussian head $\mathcal{D}$. Our stage 2 module is applied for $T = 4$ steps, starting from $z^{(0)}$: each iteration takes the tokens from the previous one, re-renders the context views, recomputes the error features, and regroups the tokens at their updated 3D positions.

2.3 PATCH-ALIGNED SUB-PIXEL GAUSSIAN RECONSTRUCTION

To obtain a dense Gaussian representation while preserving the spatial structure of image features, we predict four Gaussians for each 4×4 fine-grained token, assigning each Gaussian to one of four non-overlapping 2×2 sub-patches. Rather than constraining a Gaussian to the fixed center ray of its assigned sub-patch, we learn a two-dimensional image-space offset $\Delta = (\Delta_x, \Delta_y)$ that adaptively shifts its ray within the corresponding region. Specifically, for a sub-patch with stride $s = 2$, we bound the offset as $\delta_x = \frac{s}{2}\tanh(\Delta_x), \delta_y = \frac{s}{2}\tanh(\Delta_y)$, such that $\delta_x, \delta_y \in [-s/2, s/2]$. Thus this guarantees that each Gaussian remains within its designated 2×2 sub-patch while allowing continuous sub-pixel localization. Given the center (u_c, v_c) of each sub-patch, the displaced image coordinate is $(u, v) = (u_c, v_c) + \delta$, which is unprojected using the predicted camera intrinsics. We then transform and normalize this direction using the camera-to-world rotation, with the camera translation serving as the corresponding ray origin.

We further reduce the number of generated Gaussian primitives through simple opacity-based pruning. While more sophisticated pruning strategies could provide additional gains, we deliberately adopt this minimal scheme to isolate the effectiveness of our representation rather than optimize the pruning procedure itself. Specifically, primitives with opacity below a predefined threshold, $\tau = 0.01$, are discarded, removing Gaussians that contribute negligibly to the rendered images. This lightweight pruning further compacts the representation without introducing additional optimization.

3 EXPERIMENTS

3.1 IMPLEMENTATION DETAILS

Setup. We train a single checkpoint exclusively on DL3DV (Ling et al., 2024) and use the same checkpoint for all evaluations without dataset-specific fine-tuning. We evaluate in-domain performance on 140 DL3DV test scenes and generalization performance on 1,600 RealEstate10K (RE10K) (Zhou et al., 2018) test scenes. On DL3DV, we use $N \in \{6, 12, 24\}$ context views, selected via farthest point sampling with maximum temporal frame gaps of 50, 100, and 150, respectively. On RE10K, we use $N = 6$ context views. We further evaluate zero-shot generalization on ScanNet++ (Yeshwanth et al., 2023) using 50 indoor scenes and $N \in \{32, 64, 128\}$ context views, providing out-of-distribution settings in both camera trajectories and the number of context views. We hold out 8 target views for novel-view evaluation on DL3DV and RE10K, and 16 target views on ScanNet++. For fair comparison, each method performs encoding and rendering at its native training resolution, while all rendered outputs are resized to a common resolution of 224×224 for evaluation, following Ye et al. (2026). We report PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018). We apply evaluation-time pose alignment (EPA) before computing these metrics for methods that do not receive ground-truth camera poses as input, while methods provided with ground-truth poses are evaluated without EPA. For pose-free prediction, we follow the two-expert framework of (Jeong et al., 2026), where a geometry expert (DA3) first predicts the camera poses, which are then used to condition our *FineSplat* network for detailed Gaussian reconstruction. Further details of the evaluation setup and other details are provided in the Appendix.

Baselines. We compare against the pose-free methods AnySplat (Jiang et al., 2025), C3G (An et al., 2026), DA3 (Lin et al., 2026), YoNoSplat (Ye et al., 2026), NoDrift3R (Sun et al., 2026), 2Xplat (Jeong et al., 2026), and ZipSplat (Veicht et al., 2026), as well as NoPoSplat (Ye et al., 2025a), which additionally requires ground-truth camera intrinsics. For pose-dependent methods, we compare against MVSplat (Chen et al., 2024a), DepthSplat (Xu et al., 2025), and MVP (Kang et al., 2026b).

3.2 NOVEL VIEW SYNTHESIS

Benchmark Performance. As shown in Table 1, *FineSplat* consistently outperforms previous state-of-the-art methods in both posed and fully pose-free settings across all input-view configurations. Compared with 2Xplat, ours improves PSNR by up to $+2.88$ dB in the posed setting and $+2.62$ dB in the pose-free setting, while using approximately $4\times$ fewer Gaussians. Moreover, ours

Table 1: Novel view synthesis on DL3DV with 6, 12, and 24 input views (224×224 resolution; 252×252 results are provided in the Appendix). p and k denote the use of ground-truth poses and intrinsics, respectively. Evaluation-time pose alignment (EPA) is only applied for pose-free evaluation, while † denotes posed results that additionally use EPA. τ denotes opacity pruning threshold.

Method	p	k	6 views				12 views				24 views			
			# GS	PSNR ↑	SSIM ↑	LPIPS ↓	# GS	PSNR ↑	SSIM ↑	LPIPS ↓	# GS	PSNR ↑	SSIM ↑	LPIPS ↓
MVSplat	✓	✓	393K	22.659	0.760	0.173	786K	21.289	0.709	0.224	1.6M	19.975	0.662	0.269
DepthSplat	✓	✓	393K	23.418	0.797	0.136	786K	21.911	0.753	0.179	1.6M	20.088	0.690	0.240
YoNoSplat	✓	✓	301K	24.717	0.817	0.139	602K	23.285	0.773	0.177	1.2M	22.664	0.758	0.192
MVP	✓	✓	301K	24.314	0.798	0.181	602K	23.432	0.775	0.210	1.2M	23.106	0.768	0.221
2Xplat	✓	✓	301K	25.217	0.827	0.150	602K	24.592	0.813	0.169	1.2M	24.354	0.811	0.174
DA3	✓	✓	1.5M	19.955	0.591	0.212	3.0M	20.263	0.602	0.213	6.1M	20.527	0.626	0.212
DA3†	✓	✓	1.5M	22.968	0.790	0.157	3.0M	22.623	0.765	0.173	6.1M	22.505	0.758	0.181
ZipSplat	✓	✓	62K	23.943	0.775	0.169	124K	22.730	0.718	0.198	249K	22.536	0.715	0.201
ZipSplat†	✓	✓	62K	24.486	0.794	0.164	124K	23.555	0.754	0.189	249K	23.460	0.754	0.190
Ours	✓	✓	75K	**27.325**	**0.871**	**0.115**	151K	**27.140**	**0.868**	**0.121**	301K	**27.233**	**0.873**	**0.119**
Ours$_{\tau=0.01}$	✓	✓	68K	27.304	**0.871**	0.118	129K	27.073	0.867	0.124	239K	27.098	0.871	0.124
NoPoSplat		✓	393K	22.766	0.743	0.179	786K	19.380	0.563	0.318	1.6M	17.860	0.495	0.397
YoNoSplat		✓	301K	24.887	0.819	0.138	602K	23.149	0.758	0.183	1.2M	22.354	0.731	0.205
2Xplat		✓	301K	25.210	0.825	0.152	602K	24.337	0.798	0.177	1.2M	24.049	0.792	0.188
Ours		✓	75K	**27.256**	**0.868**	**0.117**	151K	**26.911**	**0.860**	**0.125**	301K	**26.861**	**0.859**	**0.128**
Ours$_{\tau=0.01}$		✓	68K	27.235	**0.868**	0.119	129K	26.846	0.859	0.129	239K	26.736	0.857	0.132
AnySplat			951K	19.027	0.554	0.235	1.8M	18.940	0.549	0.262	3.2M	19.703	0.596	0.249
YoNoSplat			301K	24.531	0.804	0.142	602K	22.933	0.746	0.187	1.2M	22.174	0.720	0.209
NoDrift3R			301K	24.922	0.826	0.127	602K	24.250	0.797	0.141	1.2M	24.242	0.794	0.142
2Xplat			301K	25.227	0.825	0.151	602K	24.360	0.799	0.177	1.2M	24.077	0.793	0.185
C3G			2K	18.608	0.481	0.400	2K	16.138	0.392	0.548	2K	15.055	0.358	0.601
DA3			1.5M	23.065	0.791	0.152	3.0M	22.742	0.773	0.166	6.1M	22.649	0.767	0.173
ZipSplat			62K	24.408	0.789	0.166	124K	23.497	0.749	0.192	249K	23.380	0.749	0.193
Ours			75K	**27.040**	**0.863**	**0.118**	151K	**26.812**	**0.858**	**0.125**	301K	**26.859**	**0.858**	**0.127**
Ours$_{\tau=0.01}$			68K	27.021	0.862	0.120	129K	26.752	0.857	0.128	239K	26.738	0.857	0.131

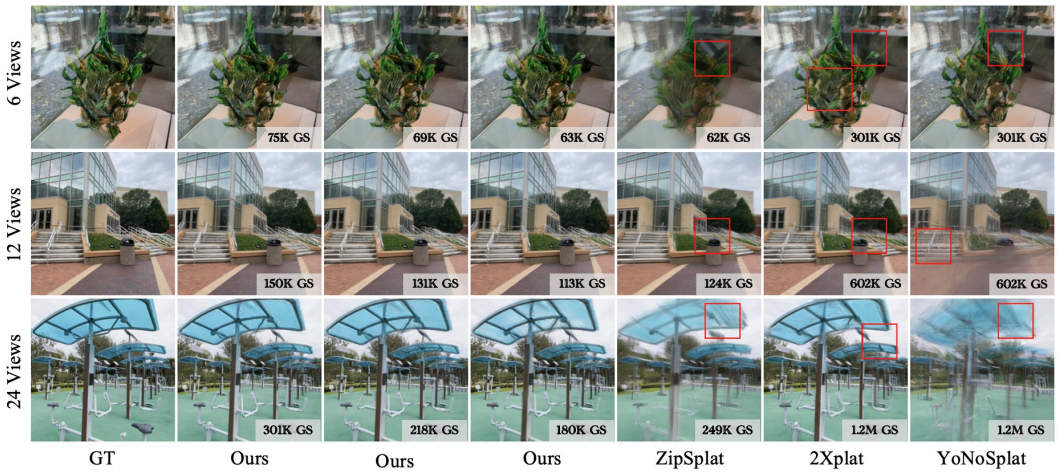

Figure 3: Qualitative comparison on DL3DV with varying numbers of input views.

remains substantially more stable as the number of views increases, which we attribute to our two-stage dual-grouping formulation with fine-grained tokens that better disambiguates redundant and highly overlapping regions across views, reducing conflicting Gaussian predictions. Moreover, unlike other methods, we achieve strong PSNR while maintaining low LPIPS, which we attribute to our ray-based formulation that directly transfers observed RGB appearance to the corresponding Gaussians, better preserving fine-grained appearance.

Cross-Dataset Generalization. We further perform zero-shot cross-domain evaluation of *FineSplat* on RE10K using the same checkpoint trained only on DL3DV. Although RE10K is not used for training, as shown in Table 2, ours consistently outperforms prior methods in the posed setting across

Table 2: Cross-dataset generalization result on RE10K with 6 input views. Evaluation-time pose alignment (EPA) is only applied for pose-free evaluation, while † denotes posed results that additionally use EPA. Methods shown in gray are trained on RE10K.

Method	p	k	6 views			
			# GS	PSNR ↑	SSIM ↑	LPIPS ↓
DepthSplat	✓	✓	393K	24.156	0.846	0.145
YoNoSplat	✓	✓	301K	25.037	0.848	0.134
2Xplat	✓	✓	301K	26.374	0.875	0.136
ZipSplat	✓	✓	62K	24.890	0.815	0.152
ZipSplat†	✓	✓	62K	26.379	0.858	0.140
DA3	✓	✓	1.5M	19.693	0.645	0.237
DA3†	✓	✓	1.5M	21.889	0.800	0.184
Ours	✓	✓	75K	**27.394**	**0.886**	**0.116**
Ours$_{\tau=0.01}$	✓	✓	67K	27.358	**0.886**	0.117
YoNoSplat			301K	24.571	0.823	0.144
NoDrift3R			301K	25.736	0.876	0.128
2Xplat			301K	27.239	0.881	0.126
C3G			2K	20.215	0.638	0.316
ZipSplat			62K	25.474	0.825	0.155
DA3			1.5M	20.631	0.712	0.228
Ours			75K	**26.769**	**0.861**	**0.125**
Ours$_{\tau=0.01}$			67K	26.738	**0.861**	0.126

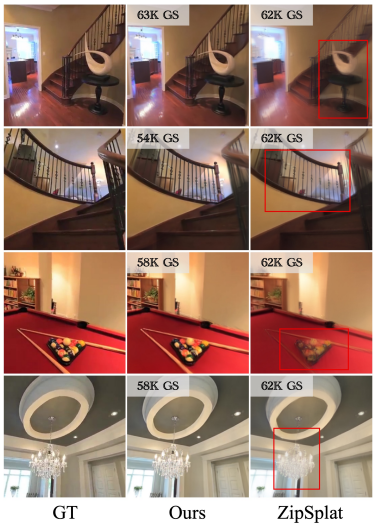

Figure 4: Qualitative comparison on RE10K with 6 input views. Ours is evaluated zero-shot without training on the RE10K dataset.

Table 3: Cross-dataset generalization results on ScanNet++. Methods shown in gray are trained on ScanNet++.

Method	32 views				64 views				128 views			
	# GS	PSNR ↑	SSIM ↑	LPIPS ↓	# GS	PSNR ↑	SSIM ↑	LPIPS ↓	# GS	PSNR ↑	SSIM ↑	LPIPS ↓
AnySplat	4.5M	16.829	0.615	0.401	8.3M	18.526	0.671	0.344	14.5M	19.397	0.697	0.321
DA3	8.1M	18.545	0.678	0.363	16.3M	19.459	0.712	0.318	32.5M	20.274	0.740	0.279
YoNoSplat	1.6M	16.886	0.600	0.432	3.2M	17.368	0.608	0.413	6.4M	17.641	0.617	0.405
NoDrift3R	1.6M	17.569	0.620	0.419	3.2M	18.935	0.668	0.348	6.4M	19.714	0.695	0.317
2Xplat	1.6M	18.043	0.651	0.402	3.2M	19.639	0.698	0.342	6.4M	20.741	0.731	0.309
C3G	2K	12.776	0.492	0.657	2K	12.595	0.482	0.660	2K	12.183	0.465	0.668
ZipSplat	332K	17.752	0.629	0.430	664K	18.468	0.651	0.399	1.3M	18.972	0.669	0.382
Ours	401K	**18.448**	**0.658**	**0.381**	803K	**20.779**	**0.729**	**0.293**	1.6M	**22.655**	**0.781**	**0.241**
Ours$_{\tau=0.01}$	296K	18.431	**0.658**	0.384	537K	20.716	**0.729**	0.299	923K	22.501	0.780	0.249

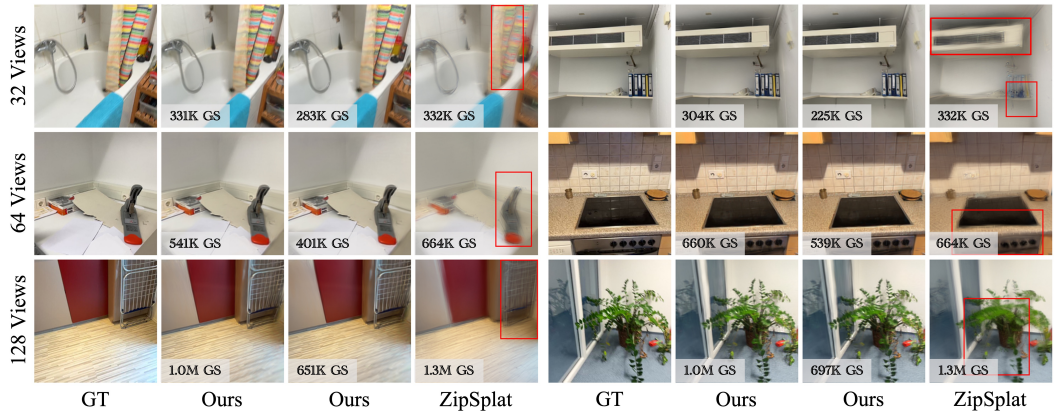

Figure 5: Qualitative comparison on ScanNet++ with varying numbers of input views.

Table 4: Effect of the number of Stage 2 steps on DL3DV with 6, 12, and 24 input views. All results are posed results using ground-truth poses and intrinsics. Step 0 denotes the initial prediction from Stage 1. The model is trained with four Stage 2 steps, and gray rows exceed this training setting.

Depth	6 views			12 views			24 views		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
0	26.109	0.852	0.133	25.794	0.846	0.141	25.776	0.850	0.141
1	26.959	0.866	0.121	26.741	0.862	0.127	26.814	0.867	0.125
2	27.201	0.870	0.117	27.005	0.866	0.123	27.097	0.872	0.121
3	27.289	**0.871**	0.116	27.100	0.867	**0.121**	27.193	**0.873**	**0.119**
4	27.325	**0.871**	**0.115**	**27.140**	**0.868**	**0.121**	**27.233**	**0.873**	**0.119**
8	**27.333**	0.870	**0.115**	27.137	0.867	0.122	27.227	0.872	0.120
16	27.143	0.865	0.120	26.783	0.857	0.130	26.796	0.860	0.131

Table 5: Masking out the views outside the view group in e_{agg} at inference time. *Within group*: e_{agg} averages only the views of the token's own view group in the first stage. *Random*: the same number of visible views is removed at random. Δ is relative to *All*. DL3DV, ground-truth poses, Ours.

e_{agg} views	6 views						12 views						24 views					
	PSNR ↑	Δ_{PSNR}	SSIM ↑	Δ_{SSIM}	LPIPS ↓	Δ_{LPIPS}	PSNR ↑	Δ_{PSNR}	SSIM ↑	Δ_{SSIM}	LPIPS ↓	Δ_{LPIPS}	PSNR ↑	Δ_{PSNR}	SSIM ↑	Δ_{SSIM}	LPIPS ↓	Δ_{LPIPS}
All (Ours)	27.325	–	0.871	–	0.115	–	27.140	–	0.868	–	0.121	–	27.233	–	0.873	–	0.119	–
Within Group	27.099	−0.226	0.866	−0.005	0.118	+0.003	26.734	−0.406	0.859	−0.009	0.128	+0.007	26.628	−0.605	0.860	−0.013	0.131	+0.012
Random	27.141	−0.184	0.867	−0.004	0.118	+0.003	26.816	−0.324	0.860	−0.008	0.126	+0.005	26.801	−0.432	0.863	−0.010	0.128	+0.009

PSNR, SSIM, and LPIPS, including methods trained directly on RE10K, while using substantially fewer Gaussians. In the fully pose-free setting, ours also achieves strong performance, demonstrating that the proposed representation transfers effectively across closely related multi-view datasets.

In addition, we assess cross-dataset generalization on ScanNet++, which presents substantial distribution shifts in both camera trajectories and the number of input views. As shown in Table 3, *FineSplat* consistently outperforms all DL3DV-trained methods across 32, 64, and 128 views, with the performance gap increasing as more views are provided. This trend highlights the scalability of our two-stage dual-grouping formulation with fine-grained tokens, which helps reduce redundancy in highly overlapping multi-view regions, as discussed above. Notably, at 64 and 128 views, ours even surpasses methods trained directly on ScanNet++, while using substantially fewer Gaussians than pixel-aligned approaches. These results demonstrate that *FineSplat* generalizes effectively to unseen scene distributions and scales robustly to substantially longer input sequences.

4 Analysis

We analyze how local interactions in 3D complement the image-space prediction of Stage 1. All results in this section are posed results on DL3DV, where the context views cover a larger portion of the scene as their number increases. We describe the components of Stage 2 in the order in which information flows, and then show how they benefit from more input views.

Stage 2 combines two error signals for each token. The self error feature e_{self} provides a direct signal from the patch where the token is predicted, while the aggregated error feature e_{agg} averages the error features over all context views in which the token is visible, regardless of its view group. Tokens then exchange these signals with their neighbors in 3D through serialized attention (Wu et al., 2024). Stage 2 consists of four attention blocks, and each block uses a different one of the four space-filling curves, so that each token meets a different set of neighbors in each block while the cost remains low. Figure 6 visualizes the window of a query token, which is a single 4×4 patch in the source view. Its neighbors come from many different frames yet lie close to the query in 3D, showing that tokens observing the same surface can exchange information across views and view groups.

Table 4 reports the rendering quality after each Stage 2 step. With fine-grained patch tokens, Stage 1 alone already achieves strong performance, providing a good initial prediction that preserves image

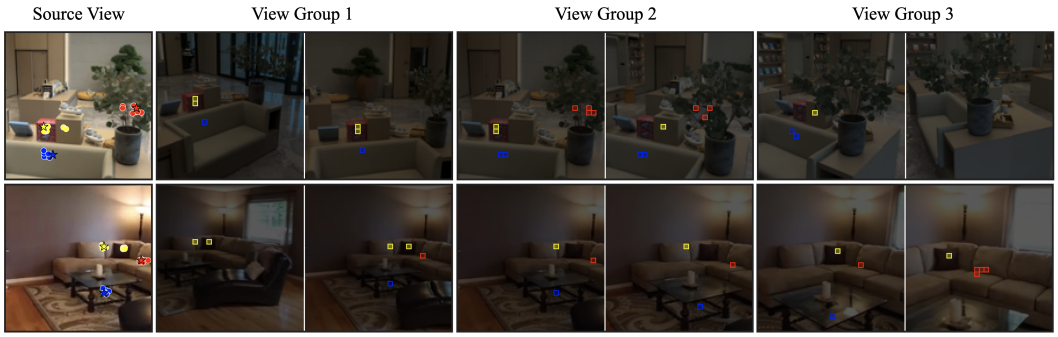

Figure 6: **Neighbors within a serialized attention window.** Tokens in the same window share a color. *Star*: query token in the reference view. *Circle*: window neighbor's 3D position projected onto the reference view. *Square*: window neighbor's 4×4 patch in its source view.

details. However, for efficiency, its fine-grained tokens interact only within local neighborhoods of views, while interactions across all views are handled at a coarser level. As a result, its quality gradually degrades from 26.11 dB with 6 views to 25.78 dB with 24 views, since a single pass must reconstruct a larger scene and tokens observing the same surface are more often separated into different view groups. Stage 2 compensates for this degradation, and as more views provide richer observations of each region, its gain increases from 1.22 dB with 6 views to 1.46 dB with 24 views. As a result, the final quality remains nearly constant across view counts. A single step already recovers about 70% of this gain, and applying more steps than the four used during training brings no additional gain.

To examine whether this growing gain comes from observations outside the view group, we restrict e_{agg} at inference time to the views within the token's own view group in Table 5. As the number of views grows, most visible views lie outside the group, from 55.5% with 6 views to 84.0% with 24 views. We therefore compare this setting with a control that removes the same number of visible views at random. Both settings discard the same number of observations, yet removing the views outside the group consistently causes a larger drop, and the difference widens with the number of views, from 0.04 dB with 6 views to 0.17 dB with 24 views. This shows that observations from other view groups carry information that the views within the group cannot provide. Together with the results in Table 4, these results indicate that FineSplat gathers observations beyond the view groups of Stage 1 and exchanges them among tokens that are close in 3D, which keeps its quality stable as the number of input views increases.

5 Conclusion and Discussion

We presented *FineSplat*, a feed-forward 3DGS framework that preserves fine-grained visual information through high-resolution patch representations while avoiding the prohibitive cost of global multi-view attention via a *dual-grouping* strategy that combines view-wise and 3D spatial locality. Together with the reconstruction feedback and compact Gaussian generation, we achieve strong novel-view synthesis quality with a small primitive budget.

While our results suggest that fine-grained multi-view interaction is an important ingredient for high-quality feed-forward reconstruction, we do not claim that fine patches are universally preferable: sufficiently large foundation models trained at scale may compensate for coarse tokenization. Nevertheless, our findings indicate that, in the multi-view transformer, revisiting token granularity and explicitly enabling fine-scale cross-view interaction can be highly effective. We also deliberately avoid relying on pretrained foundation-model features. In our preliminary experiments, combining such features with fine-grained tokens does not show any noticeable improvement. Exploring how coarse, semantically and geometrically rich foundation-model features can be more effectively integrated with fine-grained local representations, particularly to improve generalization, remains a promising direction for future work.

AI USE STATEMENT

The authors used generative AI only as a supporting tool during manuscript preparation, primarily for language polishing, improving clarity, arranging figures, and handling LaTeX formatting. All scientific ideas, methodological choices, technical developments, and experimental procedures were conceived and carried out by the authors. Any AI-assisted material was subsequently checked and revised by the authors, who assume full responsibility for the contents of the manuscript.

ETHICS STATEMENT

All experiments in this study were conducted using existing datasets that are publicly available. The study does not involve human participants or the use of personally identifiable information. We are not aware of any specific ethical issues associated with the proposed approach or its intended use.

REPRODUCIBILITY STATEMENT

The method, the implementation, training setup, and evaluation protocols are described in Sec. 3, and further details are provided in the Appendix. All datasets are publicly available, and we follow the published evaluation protocols where applicable. We will release the source code and weights.

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. C3g: Learning compact 3d representations with 2k gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

Alessandro Burzio, Tobias Fischer, Sven Elflein, Qunjie Zhou, Riccardo de Lutio, Jiawei Ren, Jiahui Huang, Shengyu Huang, Marc Pollefeys, Laura Leal-Taixé, et al. D\'ej\a view: Looping transformers for multi-view 3d reconstruction. *arXiv preprint arXiv:2605.30215*, 2026.

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 (CVPR)*, 2024.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024a.

Yun Chen, Jingkang Wang, Ze Yang, Sivabalan Manivasagam, and Raquel Urtasun. G3r: Gradient guided generalizable reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024b.

Yutong Chen, Marko Mihajlovic, Xiyi Chen, Yiming Wang, Sergey Prokudin, and Siyu Tang. Splatformer: Point transformer for robust 3d gaussian splatting. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

In Cho, Jeonghwan Cho, Mijin Yoo, Gim Hee Lee, and Seon Joo Kim. Atsplat: Compact feedforward 3d gaussian splatting with adaptive token expansion. *arXiv preprint arXiv:2607.20417*, 2026.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Roni Itkin, Noam Issachar, Yehonatan Keypur, Xingyu Chen, Anpei Chen, and Sagie Benaim. Globalsplat: Efficient feed-forward 3d gaussian splatting via global scene tokens. *arXiv preprint arXiv:2604.15284*, 2026.

Ayush Jain, Pushkal Katara, Nikolaos Gkanatsios, Adam W Harley, Gabriel Sarch, Kriti Aggarwal, Vishrav Chaudhary, and Katerina Fragkiadaki. Odin: a single model for 2d and 3d segmentation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3564–3574. IEEE, 2024.

Hwasik Jeong, Seungryong Lee, Gyeongjin Kang, Seungkwon Yang, Xiangyu Sun, Seungtae Nam, and Eunbyung Park. 2xplat: Decoupling geometry and appearance modeling for feed-forward 3d gaussian splatting. *arXiv preprint arXiv:2603.21064*, 2026.

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

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 (CVPR)*, 2026a.

Gyeongjin Kang, Seungkwon Yang, Seungtae Nam, Younggeun Lee, Jungwoo Kim, and Eunbyung Park. Multi-view pyramid transformer: Look coarser to see broader. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026b.

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

Injae Kim, Chaehyeon Kim, Minseong Bae, Minseok Joo, and Hyunwoo J. Kim. F4splat: Feed-forward predictive densification for feed-forward 3d gaussian splatting, 2026. URL `https://arxiv.org/abs/2603.21304`.

Karim Knaebel, Kadir Yilmaz, Daan De Geus, Alexander Hermans, David Adrian, Timm Linder, and Bastian Leibe. Dino in the room: Leveraging 2d foundation models for 3d segmentation. In *2026 International Conference on 3D Vision (3DV)*, pp. 1884–1894. IEEE, 2026.

Lingyu Kong, Ruicheng Li, Ruicheng Wang, Sicheng Xu, Chengtang Yao, Jianfeng Xiang, and Jiaolong Yang. Moge-3: Fine-detail monocular geometry estimation with self-guided sparse volumetric refinement. *arXiv e-prints*, pp. arXiv–2607, 2026.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. Megasam: Accurate, fast, and robust structure and motion from casual dynamic videos. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10486–10496. IEEE, 2025.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

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.

Lahav Lipson, Zachary Teed, and Jia Deng. Raft-stereo: Multilevel recurrent field transforms for stereo matching. In *2021 International conference on 3D vision (3DV)*, pp. 218–227. IEEE, 2021.

Yueh-Cheng Liu, Lukas Höllein, Matthias Nießner, and Angela Dai. Quicksplat: Fast 3d surface reconstruction via learned gaussian initialization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Yueh-Cheng Liu, Jozef Hladký, Matthias Nießner, and Angela Dai. Diff3r: Feed-forward 3d gaussian splatting with uncertainty-aware differentiable optimization. 2026. URL https://arxiv.org/abs/2604.01030.

Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pp. 674–679, 1981.

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 (CVPR)*, 2026.

Hoang Chuong Nguyen, Renjie Wu, Jose M. Alvarez, and Miaomiao Liu. Rosplat: Robust feed-forward pixel-wise gaussian splatting for varying input views and high-resolution rendering. *arXiv preprint arXiv:2605.13093*, 2026.

Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. Ecosplat: Efficiency-controllable feed-forward 3d gaussian splatting from multi-view images. In *CVPR*, 2026.

Naama Pearl, Stefano Esposito, Haofei Xu, Amit Peleg, Patricia Gschossmann, Lorenzo Porzi, Peter Kontschieder, Gerard Pons-Moll, and Andreas Geiger. Learn2splat: Extending the horizon of learned 3dgs optimization. *arXiv preprint arXiv:2605.15760*, 2026.

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 (CVPR)*, 2026.

Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 2432–2439. IEEE, 2010.

Xiangyu Sun, Liu Liu, Seungkwon Yang, Jingbing Han, Seungtae Nam, Zhizhong Su, and Eunbyung Park. Nodrift3r: Raymap-guided coupling for drift-robust unposed feed-forward 3d reconstruction. In *European Conference on Computer Vision*, pp. 440–457. Springer, 2026.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021.

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

Chen Wang, Hao Tan, Wang Yifan, Zhiqin Chen, Yuheng Liu, Kalyan Sunkavalli, Sai Bi, Lingjie Liu, and Yiwei Hu. tttlrm: Test-time training for long context and autoregressive 3d reconstruction. 2026a.

Fangjinhua Wang, Silvano Galliani, Christoph Vogel, and Marc Pollefeys. Itermvs: Iterative probability estimation for efficient multi-view stereo. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8596–8605. IEEE, 2022.

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 (CVPR)*, 2025a.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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 38*, pp. 125793–125822. Neural Information Processing Systems Foundation, Inc., 2025b. doi: 10.52202/085713-3782.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2026b.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026c.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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 (CVPR)*, 2025.

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

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 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025a.

Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. Yonosplat: You only need one model for feedforward 3d gaussian splatting. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

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

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

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19447–19456. IEEE, 2024.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

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

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 (CVPR)*, 2025.

Xiaoxue Zhang, Xiaoxu Zheng, Yixuan Yin, Tiao Zhao, Kaihua Tang, Michael Bi Mi, Zhan Xu, and Dave Zhenyu Chen. Anchorsplat: Feed-forward 3d gaussian splatting with 3d geometric priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18924–18933, June 2026a.

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

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

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 (ICCV)*, 2025.

A Related Works

Feed-forward 3DGS and compact Gaussian representation. Feed-forward 3DGS methods amortize per-scene optimization (Kerbl et al., 2023) by directly predicting Gaussians from input images in a single forward pass. One common design is pixel-aligned prediction, where each pixel is unprojected along its camera ray into one Gaussian, using epipolar attention (Charatan et al., 2024), cost volumes (Chen et al., 2024a; Xu et al., 2025), or large transformers (Zhang et al., 2024; Ziwen et al., 2025; Kang et al., 2026b; Wang et al., 2026a). This keeps the geometry well-conditioned and helps preserve fine appearance, but the number of Gaussians grows linearly with resolution and view count, often producing many redundant primitives in overlapping regions (Park et al., 2026; Kim et al., 2026; Nguyen et al., 2026). In contrast, Non-pixel-aligned methods predict a compact set of Gaussians from learnable tokens, clustered scene features, 3D anchors, or voxels (Ren et al., 2026; An et al., 2026; Veicht et al., 2026; Wang et al., 2026b; Moreau et al., 2026; Zhang et al., 2026a; Itkin et al., 2026; Zhang et al., 2026b; Wang et al., 2025b), but lose the explicit pixel-ray correspondence, which tends to hurt fine detail and makes optimization harder. Between these extremes, patch-aligned methods predict Gaussians at a reduced spatial resolution (Kang et al., 2026a; Xu et al., 2026). In our framework, we extend this patch-aligned approach by generating only a small number of Gaussians along the rays of each sub-patch and further compacting the representation through simple opacity-based pruning. This deliberately simple strategy proves highly effective, offering a strong alternative to recent non-pixel-aligned methods.

Pose-free feed-forward 3DGS and geometry foundation models. Geometry foundation models (Wang et al., 2024; Leroy et al., 2024; Wang et al., 2025a; 2026c; Lin et al., 2026) estimate camera poses and dense geometry directly from unposed images. Building on such models, methods for the unposed setting jointly predict cameras and Gaussians (Ye et al., 2025a; Zhang et al., 2025; Jiang et al., 2025; Ye et al., 2026), typically in a pixel-aligned manner on top of the geometry backbone. However, these models are typically built on relatively coarse image tokens, which are well suited for global geometric reasoning but may discard local appearance cues important for fine-grained Gaussian reconstruction. Existing pose-free 3DGS methods often fine-tune geometry foundation models end-to-end so that the pretrained features can adapt to appearance reconstruction and recover task-specific visual cues. However, this end-to-end adaptation is computationally expensive, and the model still operates on representations shaped by the pretrained backbone's coarse tokenization, which may limit its ability to fully recover fine-grained appearance information. Recent works explore decoupling geometry estimation from appearance modeling (Zhang et al., 2025; Jeong et al., 2026) and we follow this decoupled design for pose-free reconstruction, using DA3 as the geometry expert while *FineSplat* performs fine-grained Gaussian prediction.

Combining image-space and 3D locality. Many useful cues for 3D reconstruction, such as dense image features and rendering errors, are naturally defined in image space, where they are grouped by proximity in the image plane rather than in the underlying scene. Several works therefore bring such image-space cues into 3D and process them according to 3D spatial locality. ODIN (Jain et al., 2024) alternates 2D within-view layers with 3D cross-view attention over unprojected features, and DITR (Knaebel et al., 2026) projects DINOv2 features onto 3D points and injects them into a Point Transformer V3 (Wu et al., 2024) backbone. MoGe-3 (Kong et al., 2026) lifts monocular geometry into a sparse voxel volume guided by image features to recover sharp details near depth boundaries. In feed-forward 3DGS, ATSplat (Cho et al., 2026) unprojects patch features into sparse 3D anchors that gather context from their 3D neighbors, and expands them where rendering errors are predicted to be high. Our second stage lifts fine tokens from all views into 3D and applies serialized attention over their 3D neighborhoods, so that tokens separated by view grouping can interact.

Iterative refinement. Iterative minimization of an objective has long been a standard approach to solving vision problems (Lucas & Kanade, 1981; Sun et al., 2010), and learning-based methods have sought to emulate this process by repeatedly updating an initial estimate with a shared network (Andrychowicz et al., 2016; Teed & Deng, 2020). Such learned iterative refinement often outperforms single-step regression and has been successfully applied to optical flow (Teed & Deng, 2020), stereo matching (Lipson et al., 2021), SLAM (Teed & Deng, 2021), structure-from-motion (Li et al., 2025), multi-view stereo (Wang et al., 2022), and multi-view 3D reconstruction (Burzio et al., 2026). For Gaussian splatting, prior works refine Gaussians with a single-step

or multiple-step network (Chen et al., 2025; Kang et al., 2026a), learn updates from rendering gradients (Chen et al., 2024b; Liu et al., 2025; 2026; Pearl et al., 2026) or rendering errors (Xu et al., 2026), or fine-tune Gaussian tokens for each scene (Ren et al., 2026).

B Additional Implementation Details

B.1 Architecture

FineSplat builds upon the two-expert design of Jeong et al. (2026), separating geometry estimation from appearance modeling. The geometry backbone is initialized from DA3-Giant (Lin et al., 2026), which consists of 40 transformer layers with a hidden dimension of 1536 and 24 attention heads, each with a head dimension of 64. It estimates the camera trajectories that are subsequently used to condition the appearance and refinement stages. The appearance module is based on MVP (Kang et al., 2026b) and adopts a 14-layer hierarchical transformer initialized with 4×4 image patches. The first two layers operate on fine-resolution tokens with a feature dimension of 256 using self-attention. The following four layers process intermediate-resolution tokens with a feature dimension of 512 using self-attention and view-wise group attention. The final eight layers operate on coarse-resolution tokens with a feature dimension of 1024 using self-attention and global attention. All attention modules use a head dimension of 64. Stage 2 operates on the 4×4 patch tokens produced by the DPT head of the appearance module. The error feature map is computed at the same $\frac{H}{4} \times \frac{W}{4}$ resolution and has 256 channels: a 128-channel error branch, which combines the feature difference of a frozen ImageNet-pretrained ResNet-18 (He et al., 2016) (conv1, layer1, and layer2 features, 256 channels in total) with the space-to-depth pixel error, and a 128-channel image branch that projects the 256-dimensional finest-level features. For each token, the error features of its source view and the mean over the views passing the frustum and occlusion tests are concatenated and projected to 256 channels. The 1024-dimensional token is projected to 512 channels, concatenated with its error feature, and fused into a 512-dimensional point feature. The 3D grouping backbone follows Point Transformer V3 (Wu et al., 2024) and consists of 4 pre-norm transformer blocks with a feature dimension of 512 and an MLP ratio of 4. For efficiency, attention is computed in a 256-dimensional subspace with 16 heads of dimension 16. Points are voxelized on a 384^3 grid over the scene bounding box and serialized along four space-filling curves. Z-order, Hilbert, and their transposed variants, one per block. Local attention is computed within groups of $S = 8$ consecutive points. A linear layer maps the output back to a 1024-dimensional token residual to output the offset of GS tokens. Stage 2 has 13.2M trainable parameters and is applied for $T = 4$ iterations with shared weights.

B.2 Training

We train FineSplat in three phases: Stage 1 is trained first, Stage 2 is then trained on top of the frozen Stage 1, and finally the whole model is fine-tuned end-to-end. In all phases, each training scene samples its number of input views from $\{6, 12, 24, 32\}$ with $\{6, 6, 6, 1\}$ held-out target views, respectively, and the per-GPU batch is $\{4, 2, 1, 1\}$ scenes.

Stage 1. We largely follow the training strategy of Jeong et al. (2026), with the key difference that FineSplat adopts a compact patch-aligned Gaussian representation. The appearance branch is trained with pixel-aligned Gaussian prediction for 120K iterations, after which the pixel-aligned decoder is replaced with our patch-aligned Gaussian decoder and the model is fine-tuned for an additional 80K iterations. The remaining setup, including the camera losses on the predicted poses and intrinsics, follows Jeong et al. (2026).

Stage 2. With Stage 1 frozen, we train Stage 2 using ground-truth camera poses and intrinsics, so that it learns to correct the Gaussians independently of pose errors. Every refinement iteration is supervised: the loss is the rendering loss on the target views averaged over the $T = 4$ iterations, and the tokens are detached between iterations so that each iteration learns to improve the state it receives. The rendering loss combines MSE and a VGG-19 perceptual loss with weights 1.0 and 0.2, identical to Stage 1. Stage 2 is trained for 20K iterations on 4 GPUs ($\{16, 8, 4, 4\}$ scenes per iteration), equivalent in sample budget to 10K iterations on 8 GPUs.

End-to-end fine-tuning. Finally, we unfreeze all parameters, including the geometry backbone and the Gaussian head, and fine-tune the full model in the fully unposed setting: the cameras used to encode the inputs, lift the tokens, compute the error maps, and render the targets are all predicted by the geometry backbone. The loss is the same rendering loss, averaged over the initial prediction and all T iterations, so that the Stage 1 prediction remains directly supervised; no camera loss is used. We fine-tune for 10K iterations on 8 GPUs ($\{32, 16, 8, 8\}$ scenes per iteration) and evaluate with an exponential moving average of the weights (decay 0.999). To fit the geometry backbone in memory, we apply gradient checkpointing to every transformer block and predict all cameras in a single joint pass during training.

B.3 Optimization

Stage 1 is optimized using AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 0.05. We set the base learning rate to 2×10^{-5} and decay it to zero using a cosine schedule. We apply gradient clipping with a maximum norm of 1.0 and use bfloat16 mixed-precision training.

Stage 2 and the end-to-end fine-tuning both use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 0.05 applied to multi-dimensional parameters only, gradient clipping at 1.0, and bfloat16 mixed precision. For Stage 2, we use a learning rate of 1×10^{-4} with a 200-iteration linear warmup, keep it constant for the first 80% of training, and then decay it to zero with a cosine schedule over the remaining 20%. For end-to-end fine-tuning, we use a single learning rate of 2×10^{-5} for all parameters with a 200-iteration linear warmup, followed by cosine decay to zero.

B.4 Details of Occlusion Test during aggregating error features

To build e_{agg}, a token must only gather error from the views that actually observe the surface it represents. If a view sees a different surface in front of the token, the error at the projected pixel belongs to that occluding surface. Averaging such error would attribute a mistake to the wrong token. We therefore select views with a frustum test and a depth test before averaging. Both tests reuse the same context cameras that place the Gaussians—ground-truth poses and intrinsics in the posed setting, and DA3 predictions in the pose-free setting—so no additional ground-truth camera is required, and the projection stays self-consistent even when the predicted cameras are inaccurate.

Frustum test. Each token k is represented by $\mathbf{x}_k$, the mean position of its $2{\times}2$ Gaussians. For every context view v we transform $\mathbf{x}_k$ into the camera frame, which gives depth $z_{k,v}$. We then project it with the intrinsics to a pixel $\mathbf{u}_{k,v} = (u, v)$. The view passes the test if the point lies in front of the camera ($z_{k,v} > 0$) and $\mathbf{u}_{k,v}$ falls inside the image ($0 \le u < W$, $0 \le v < H$).

Depth test. For each view that passes the frustum test, we render the expected depth D_v from the current Gaussians. This is the same render that produces the error map E_v, so it is refreshed at every refinement pass. We sample D_v bilinearly at $\mathbf{u}_{k,v}$ and keep the view if

$$z_{k,v} \le (1 + \epsilon)\, D_v(\mathbf{u}_{k,v}), \qquad \epsilon = 0.15. \tag{6}$$

In words, the token may be at most 15% behind the rendered surface. We use a relative tolerance for two reasons. First, $\mathbf{x}_k$ is the mean of four Gaussians rather than a point on the surface. Second, the expected depth is smoothed at depth discontinuities. A relative tolerance absorbs both effects at every scene scale. Pixels that no Gaussian covers have no occluder, and the view is kept.

Examples of views that fail the frustum test or the depth-based occlusion test are shown in Figure 7.

C Evaluation Details

C.1 Evaluation-Time Pose Alignment

Pose-free reconstruction is inherently ambiguous, as multiple 3D scenes and camera configurations can explain the same unposed inputs. Thus, the reconstructed scene may not be perfectly aligned with the ground-truth evaluation frame. Following prior pose-free protocols (Wang et al., 2021; Fan et al., 2024; Ye et al., 2025a), we freeze the reconstructed 3D Gaussians and optimize each target camera pose for 200 iterations using an ℓ_1 photometric loss, optionally combined with LPIPS. This

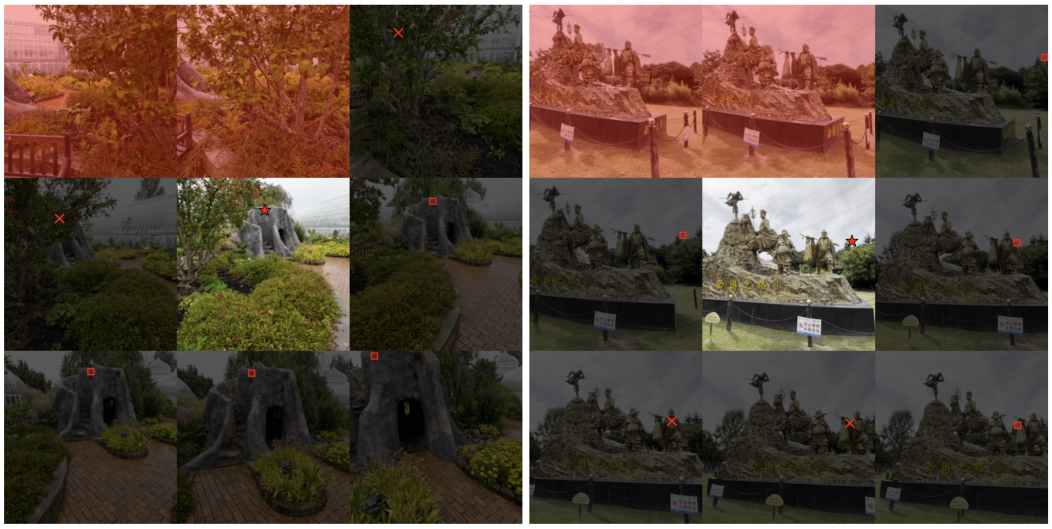

Figure 7: **View selection for e_{agg}.** The query token (*star*) in the source view (center) is projected into the other context views. *Squares* mark views that pass both the frustum and depth tests and are included in e_{agg}. *Crosses* mark views that fail the depth test, and *red-tinted views* fail the frustum test.



Figure 8: Resize filter comparison.

evaluation-time pose alignment is used only for pose-free settings. When ground-truth camera poses are available, we directly use them without additional alignment, following YoNoSplat.

C.2 Resize Filter

As noted in ZipSplat, the choice of resizing filter used to downsample rendered images and ground-truth target images can noticeably affect the reported evaluation metrics. ZipSplat adopts bilinear interpolation with anti-aliasing. In our evaluation, we instead use PIL Lanczos as the default resizing filter, as it better preserves high-frequency image details and visual fidelity during downsampling (Fig. 8), allowing the evaluation to more faithfully reflect the fine-grained appearance details preserved by the predicted Gaussians. Accordingly, unless otherwise specified, all evaluation metrics reported in this paper are computed using PIL Lanczos resizing.

C.3 Evaluation Resolution

The compared methods are trained at different input resolutions: MVSplat at 256×256, DepthSplat at 256×448, C3G, YoNoSplat, MVP, NoDrift3R, and 2Xplat at 224×224, ZipSplat at 252×252,

Table 6: Novel view synthesis on DL3DV under the evaluation protocol of ZipSplat (Veicht et al., 2026). Evaluation-time pose alignment (EPA) is only applied for pose-free evaluation, while † denotes posed results that additionally use EPA.

Method	p	k	6 views				12 views				24 views			
			# GS	PSNR ↑	SSIM ↑	LPIPS ↓	# GS	PSNR ↑	SSIM ↑	LPIPS ↓	# GS	PSNR ↑	SSIM ↑	LPIPS ↓
MVSplat	✓	✓	393K	22.66	0.760	0.173	786K	21.29	0.709	0.224	1.6M	19.98	0.662	0.269
DepthSplat	✓	✓	393K	23.42	0.797	0.136	786K	21.91	0.753	0.179	1.6M	20.09	0.690	0.240
DA3	✓	✓	1.5M	23.99	0.805	0.158	3.0M	22.84	0.758	0.190	6.1M	21.70	0.710	0.230
ZipSplat†	✓	✓	62K	25.34	0.810	0.169	124K	24.37	0.773	0.194	249K	24.23	0.773	0.194
Ours	✓	✓	95K	28.23	0.884	0.113	191K	28.07	0.882	0.119	381K	28.11	0.886	0.119
NoPoSplat†		✓	393K	22.77	0.743	0.179	786K	19.38	0.563	0.318	1.6M	17.86	0.495	0.397
Ours		✓	95K	28.01	0.875	0.119	191K	27.80	0.873	0.125	381K	27.66	0.872	0.130
AnySplat			951K	21.70	0.725	0.187	1.8M	21.01	0.687	0.220	3.2M	20.74	0.669	0.236
C3G			2K	18.70	0.492	0.409	2K	16.50	0.421	0.534	2K	15.17	0.376	0.583
DA3			1.5M	23.77	0.795	0.165	3.0M	22.38	0.736	0.208	6.1M	21.69	0.711	0.229
YoNoSplat			301K	24.10	0.783	0.160	602K	22.73	0.736	0.200	1.2M	22.01	0.710	0.223
ZipSplat			62K	25.24	0.804	0.172	124K	24.27	0.767	0.197	249K	24.14	0.768	0.198
Ours			95K	27.26	0.859	0.123	191K	26.77	0.849	0.132	381K	27.05	0.856	0.134

Table 7: Ablation of view-wise grouping with different group sizes on DL3DV using 6, 12, and 24 input views. To isolate the effect of view-wise grouping, we disable spatial grouping and vary only the view-group size, with a group size of 2 used by default during training.

View-group size	6 views			12 views			24 views		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
1	22.026	0.776	0.186	22.999	0.793	0.178	23.727	0.811	0.168
2	26.094	0.850	0.135	25.808	0.844	0.143	25.718	0.847	0.144
4	–	–	–	25.501	0.838	0.148	25.395	0.842	0.148
6	25.573	0.835	0.147	25.173	0.830	0.155	24.987	0.834	0.155
12	–	–	–	24.549	0.810	0.170	24.152	0.813	0.173
24	–	–	–	–	–	–	23.435	0.788	0.194

and AnySplat at 448×448. To ensure a fair and meaningful comparison, we perform Gaussian reconstruction and rendering at each method's native training resolution, while resizing only the final rendered and ground-truth target images to a common resolution of 224×224 for metric computation, following the evaluation protocol of YoNoSplat. Since our method operates natively at 224×224, it uses the smallest input resolution among the compared methods. Thus, this protocol does not provide our method with an advantage from a larger input resolution or field of view.

C.4 Comparison under the ZipSplat Protocol

To directly compare with the metrics reported in the original ZipSplat paper (Veicht et al., 2026), we additionally evaluate our model under the same experimental protocol. Specifically, we fine-tune our model at a resolution of 252×252, perform both reconstruction and rendering at this resolution, and resample images with bilinear interpolation with anti-aliasing when computing the evaluation metrics, following ZipSplat. The resulting comparison is reported in Table 6.

D Additional Analysis

D.1 View-wise Token Grouping with Different Sizes

We ablate the view-group size while disabling spatial grouping to isolate the effect of view-wise token grouping in Table 7. Using a group size of 1, which reduces to independent per-view self-attention, substantially degrades performance across all input-view settings. A group size of 2 con-

Table 8: Ablation on the error feature on DL3DV with 6, 12, and 24 input views. e_{self} and e_{agg} denote the error features sampled at the token's own patch and averaged over the views in which the token is visible, respectively. All variants concatenate the 4×4 patch feature from Stage 1 to the error features. All results are posed results after the fourth Stage 2 step, and Δ is relative to the default.

	e_{self}	e_{agg}	6 views			12 views			24 views		
			PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Value	✓		27.111	0.867	0.119	26.914	0.863	0.126	26.913	0.866	0.126
		✓	26.821	0.862	0.123	26.554	0.858	0.131	26.547	0.862	0.130
	✓	✓	**27.216**	**0.869**	**0.118**	**27.047**	**0.865**	**0.124**	**27.102**	**0.870**	**0.123**
Δ	✓		−0.105	−0.0020	+0.0014	−0.133	−0.0027	+0.0022	−0.189	−0.0037	+0.0030
		✓	−0.395	−0.0063	+0.0057	−0.493	−0.0077	+0.0070	−0.555	−0.0081	+0.0075
	✓	✓	–	–	–	–	–	–	–	–	–

sistently performs best, as it matches the grouping strategy used during training. Increasing the group size beyond 2 leads to only moderate degradation. Larger groups still contain the pairwise interactions learned with group size 2, while introducing additional cross-view interactions that act as increasing noise as the group size grows.

D.2 Error Feature Composition

We examine the contribution of the two error features in Table 8. Unlike our main model, which further fine-tunes the whole framework end-to-end after training Stage 2, all variants in this ablation are trained only up to Stage 2 on top of a frozen Stage 1 with ground-truth poses, differing only in the composition of the error features. Therefore, the absolute numbers differ slightly from those in Table 1. Removing either feature degrades the quality in all settings, indicating that the two features provide complementary information. Removing e_{self} causes the largest drop, from 0.40 dB with 6 views to 0.56 dB with 24 views in PSNR. Since e_{self} is sampled at the patch where the token is predicted, it provides the most direct signal of how well the token explains its own observation, and the model relies on it most heavily. Removing e_{agg} leads to a smaller but consistent drop, which grows with the number of views from 0.11 dB to 0.19 dB. This suggests that e_{agg} becomes more useful as each token is observed by more views, which we examine further below.

E ADDITIONAL QUALITATIVE RESULTS

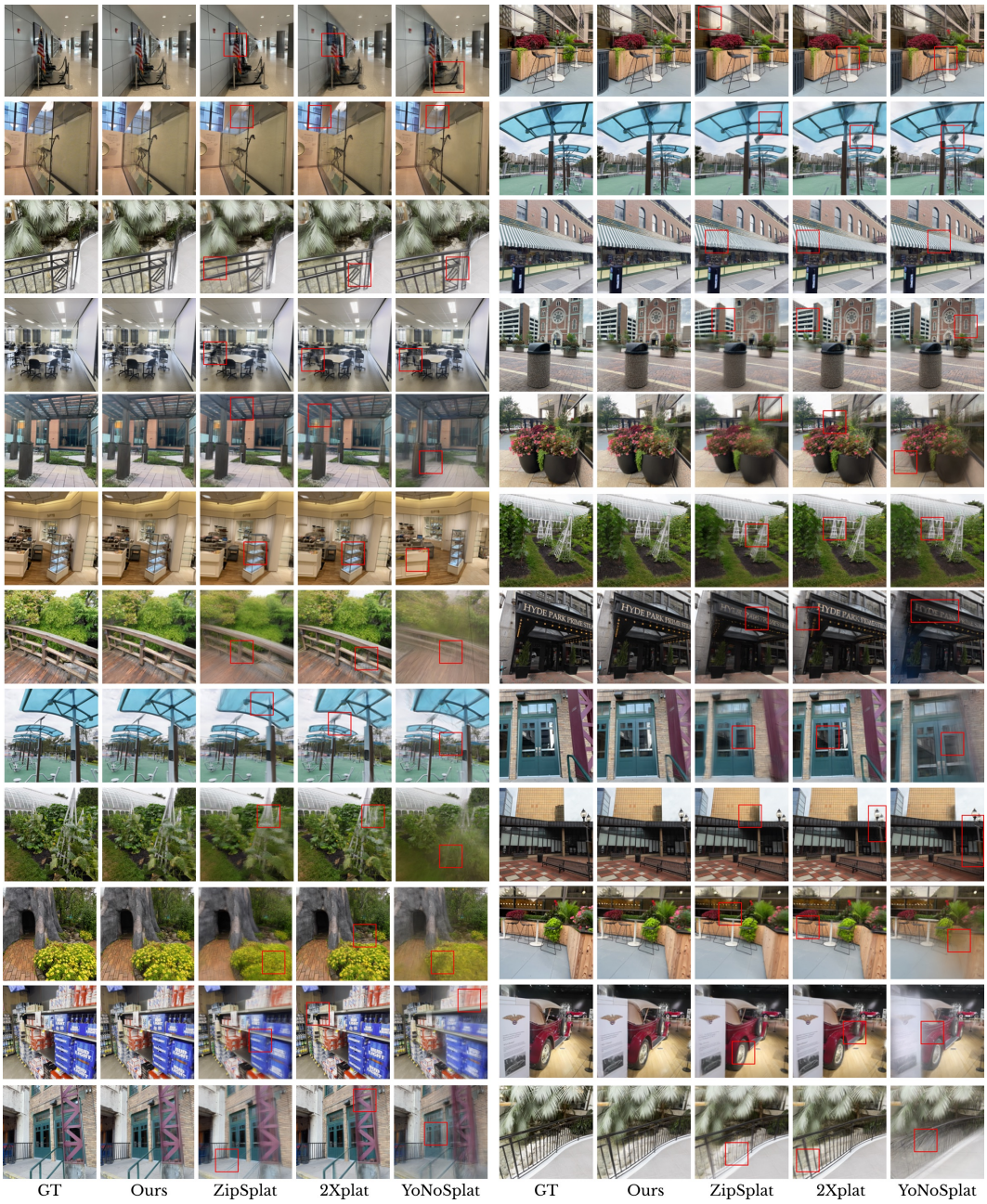

Figure 9: Qualitative comparison on DL3DV (224 × 224 resolution, 1-4 rows: 6 context views, 5-8 rows: 12 context views, 9-12 rows: 24 context views).

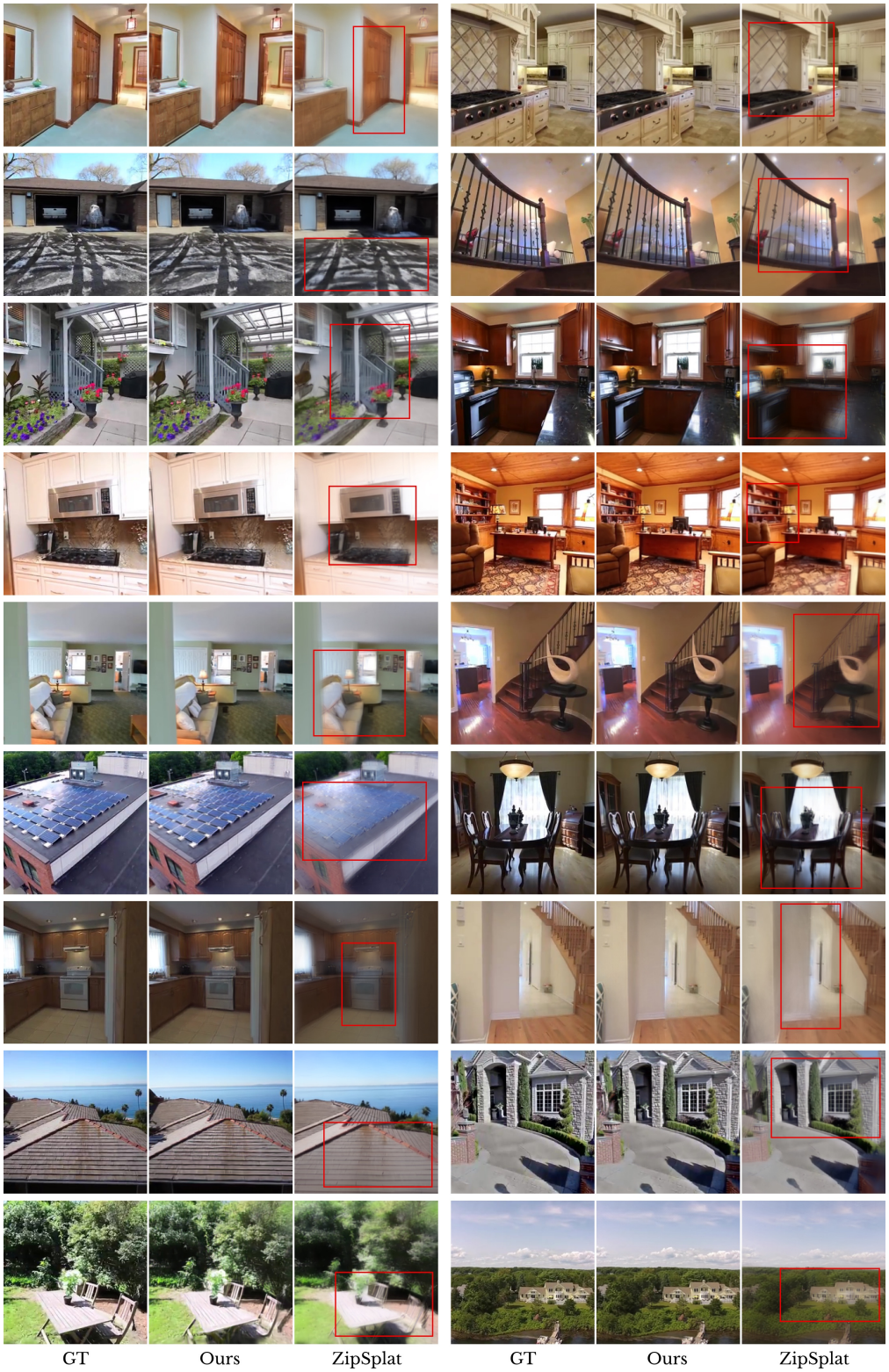

Figure 10: Qualitative comparison on RE10K at 224×224 resolution with 6 input views. Our model is evaluated zero-shot without training on RE10K, whereas ZipSplat is trained on RE10K.

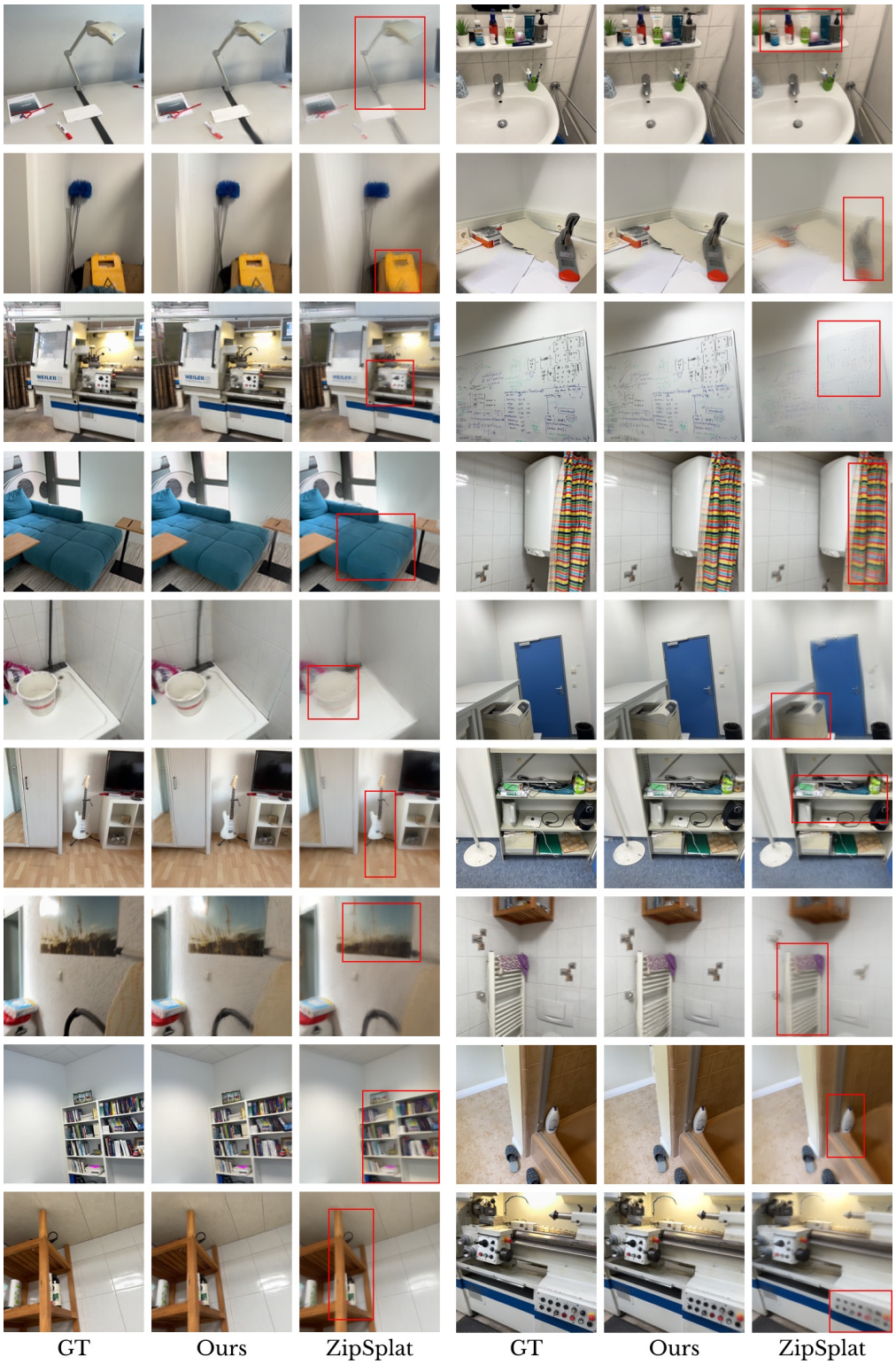

Figure 11: Qualitative cross-dataset generalization results from DL3DV to ScanNet++ (224 × 224 resolution, 1-3 rows: 32 context views, 4-6 rows: 64 context views, 7-9 rows: 128 context views).