ATSPLAT: COMPACT FEED-FORWARD 3D GAUSSIAN SPLATTING WITH ADAPTIVE TOKEN EXPANSION

Anonymous authors
Paper under double-blind review

ABSTRACT

3D Gaussian Splatting (3DGS) achieves high-quality novel-view synthesis by optimizing freely placed primitives in 3D and adaptively densifying them in under-reconstructed regions. However, this scene-adaptive capacity allocation is largely lost in existing feed-forward 3DGS methods, where primitive counts and placement depend on input image grids rather than scene complexity. We present ATSplat, a feed-forward 3DGS framework that restores scene-adaptive capacity allocation through Adaptive 3D Tokens. ATSplat first lifts coarse patch-level depth and camera cues into sparse 3D anchor tokens, forming a compact scene scaffold. The tokens are progressively and selectively expanded and refined during decoding, guided by an Adaptive Token Expansion module that predicts per-token expansion scores at each stage. The expansion scores are supervised by the rendering error reduction associated with token expansion. Each refined token is then decoded into local Gaussians with learnable 3D offsets, decoupling primitive placement from input image grids. Experiments on the RealEstate10K and DL3DV datasets highlight that ATSplat achieves competitive rendering quality while using up to $61\times$ fewer Gaussians than dense pixel-aligned 3DGS methods.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) has become a powerful representation for novel-view synthesis, achieving high-quality reconstruction with real-time rendering. The strength of this pipeline lies in the flexibility of its representation: starting from a sparse point cloud as a coarse initialization, adaptive densification progressively redirects representational capacity and computation toward under-reconstructed regions. This sparse-to-adaptive capacity allocation based on reconstruction difficulty established 3DGS as a leading representation for novel-view synthesis.

Yet these advantages are at the cost of per-scene optimization, limiting the practicality of 3DGS. A growing body of work therefore seeks to bypass this cost, by reconstructing Gaussian primitives directly in a single forward pass. The dominant design in these feed-forward 3DGS frameworks is a pixel-aligned formulation (Charatan et al., 2024; Chen et al., 2024a; Xu et al., 2025; Kang et al., 2026a): one Gaussian is regressed at every input pixel and lifted along its camera ray, reducing primitive placement to depth estimation. While this formulation simplifies the reconstruction problem, these methods often suffer from significant rendering inefficiency and storage costs as the number of primitives grows with both image resolution and the number of input views.

More fundamentally, this inefficiency arises from how representation capacity is parameterized and allocated. By tying Gaussians to input image grids, pixel-aligned frameworks make primitive placement and capacity allocation depend on camera sampling rather than reconstruction difficulty. As a result, they sacrifice a key strength of 3DGS optimization framework: the ability to allocate capacity adaptively according to scene complexity. Closing this gap calls for a feed-forward design in which capacity allocation reflects scene complexity instead of image structure.

We present ATSplat, a feed-forward 3DGS framework that restores scene-adaptive capacity allocation of 3DGS with Adaptive Tokens. Our framework reinterprets the sparse-to-adaptive principle of 3DGS optimization as feed-forward operations centered on adaptive anchor tokens. ATSplat first lifts coarse patch-level depth and camera information into sparse 3D anchor tokens, forming a compact scene scaffold. Tokens in under-reconstructed regions are progressively identified, expanded, and refined during decoding, concentrating representational capacity and computation where the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 1: ATSplat reconstructs compact 3D Gaussians from multi-view captures in a single forward pass. Unlike dense pixel-aligned feed-forward methods that place Gaussians according to input image grids, ATSplat starts from sparse 3D anchor tokens and progressively expands and refines tokens expected to benefit from additional capacity. This sparse-to-adaptive allocation preserves fine structures with a compact Gaussian set, enabling high-quality novel-view rendering. With 16 images at 540×960 resolution, ATSplat completes reconstruction in 0.4 seconds with 358K Gaussians.

scene demands it. The refined tokens guide expansion at each stage, allowing capacity allocation to adapt as reconstruction progresses. Each token is finally decoded into a small set of local Gaussians with relative 3D offsets from its anchor, decoupling primitive placement from input image grids.

The central challenge in adaptive expansion is identifying under-reconstructed regions that would benefit from additional capacity at each decoding stage. Unlike in per-scene 3DGS optimization, gradient statistics accumulated over multiple optimization steps are not directly available in a single forward pass. Our Adaptive Token Expansion module addresses this by learning a per-token expansion score, supervised by the rendering error reduction associated with token expansion. This supervision guides ATSplat to concentrate additional capacity in regions where expansion is expected to reduce reconstruction error, avoiding unnecessary expansion where little gain is expected.

Thanks to its progressive token expansion and flexible anchor design, ATSplat uses its representational budget more efficiently than dense pixel-aligned feed-forward 3DGS methods. Experiments on two representative datasets, RealEstate10K (Zhou et al., 2018) and DL3DV (Ling et al., 2024), demonstrate that ATSplat achieves competitive rendering quality while using up to $61\times$ fewer Gaussians than dense pixel-aligned 3DGS methods. By restoring the sparse-to-adaptive principle of 3DGS, ATSplat demonstrates that high-quality feed-forward 3DGS arises from how representation capacity is allocated, not from how densely it is sampled.

2 Related Work

3D representations. Neural radiance fields (NeRF) represent a scene as a continuous radiance-density field and synthesize novel views through volumetric rendering (Mildenhall et al., 2020). Despite its high fidelity, NeRF requires dense sampling along camera rays for volumetric rendering,

making novel-view synthesis computationally expensive. 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) instead represents scenes with anisotropic Gaussian primitives and enables high-quality real-time rendering through differentiable rasterization. A rapidly growing body of follow-up work has extended it along many directions, including surface reconstruction (Huang et al., 2024; Guédon & Lepetit, 2024; Yu et al., 2024b), anti-aliased rendering (Yu et al., 2024a), dynamic scene modeling (Wu et al., 2024; Yang et al., 2024), and compact, structured representations (Lu et al., 2024).

Feed-forward 3D Gaussian Splatting. Instead of optimizing Gaussians per scene, recent feed-forward 3DGS methods directly predict Gaussian primitives from multi-view captures. Splatter Image (Szymanowicz et al., 2024) demonstrates single-view object reconstruction with Gaussian maps, while pixelSplat predicts 3D Gaussians from posed image pairs by lifting pixel-aligned predictions along camera rays (Charatan et al., 2024). MVSplat (Chen et al., 2024a) further improves this feed-forward framework with a plane-sweep cost volume, and DepthSplat (Xu et al., 2025) connects multi-view depth estimation with Gaussian prediction using depth priors. Subsequent works extend feed-forward 3DGS frameworks to 360-degree (Chen et al., 2024b), unposed (Hong et al., 2025), and unconstrained settings (Jiang et al., 2025). Large reconstruction models (LRM) further scale this pixel-aligned paradigm with large transformer backbones (Zhang et al., 2024a; Kang et al., 2026a;b). Although these methods achieve impressive reconstruction speed and quality, most follow dense pixel-aligned formulations that couple primitive count to image resolution and view count, producing redundant Gaussians in simple and textureless regions.

Compact Feed-forward 3D Gaussian Splatting. Several recent works attempt to address the limitation of dense pixel-aligned Gaussian predictions. FreeSplat and FreeSplat++ fuse overlapping pixel-aligned Gaussians across views and remove redundant or floating primitives for indoor 3D scene reconstruction (Wang et al., 2024; 2026c). Gaussian Graph Network models cross-view relations among Gaussian groups and pools them into a more efficient representation (Zhang et al., 2024b), while Fuse-and-Refine aggregates pixel-aligned primitives into a canonical 3D space before refinement (Wang et al., 2025b). Generative Densification (Nam et al., 2025) takes a complementary direction: rather than reducing primitives, it learns a feed-forward densification module that upsamples features to generate fine Gaussians for high-frequency details. These methods mostly operate on top of reconstructed pixel-aligned Gaussians, and introduce post-hoc processing.

An emerging line of work moves more directly beyond dense pixel-aligned formulations. This includes decoupling Gaussian prediction from pixels using voxel-aligned prediction (Wang et al., 2026b), 3D anchored feature volumes (Zhang et al., 2026a), global learnable tokens (An et al., 2026; Ren et al., 2026; Itkin et al., 2026), or adaptive Gaussian locations within image grids (Moreau et al., 2026). Other methods reduce Gaussian counts through spatial downsampling or adaptive sampling from pixel-aligned predictions. iLRM (Kang et al., 2026a) predicts a more compact set of Gaussians in a separate, low-resolution grid, and ReSplat (Xu et al., 2026) initializes compact Gaussians from spatially downsampled image grids and recurrently refines them using rendering-error feedback. F4Splat selects its final Gaussians from multi-scale maps using gradient-supervised scores (Kim et al., 2026), and SparseSplat (Zhang et al., 2026b) adaptively samples initial pixel-aligned Gaussian locations based on image entropy and refines their features through a point transformer.

ATSplat is aligned with this emerging direction, but differs in how capacity is adaptively allocated. Instead of uniformly downsampling or adaptively sampling pixel-aligned predictions, ATSplat constructs scene-conditioned sparse 3D anchor tokens from coarse depth and cameras. The tokens that would benefit from additional capacity are progressively expanded and refined during decoding. This sparse-to-adaptive formulation fills the missing component in feed-forward 3DGS: progressive capacity allocation according to scene complexity, rather than image structure.

3 METHOD

3.1 PROBLEM FORMULATION

We aim to reconstruct a 3D Gaussian representation of a scene from posed multi-view images in a single forward pass, avoiding the costly per-scene optimization required by standard 3DGS. Given V posed images $\{\mathbf{I}_v, \boldsymbol{\pi}_v\}_{v=1}^{V}$, a feed-forward 3DGS model predicts a set of Gaussian attributes:

$$f_\theta : \{\mathbf{I}_v, \boldsymbol{\pi}_v\}_{v=1}^{V} \longmapsto \{(\boldsymbol{\mu}_g, \mathbf{q}_g, \mathbf{s}_g, \alpha_g, \mathbf{SH}_g)\}_{g=1}^{G}, \tag{1}$$

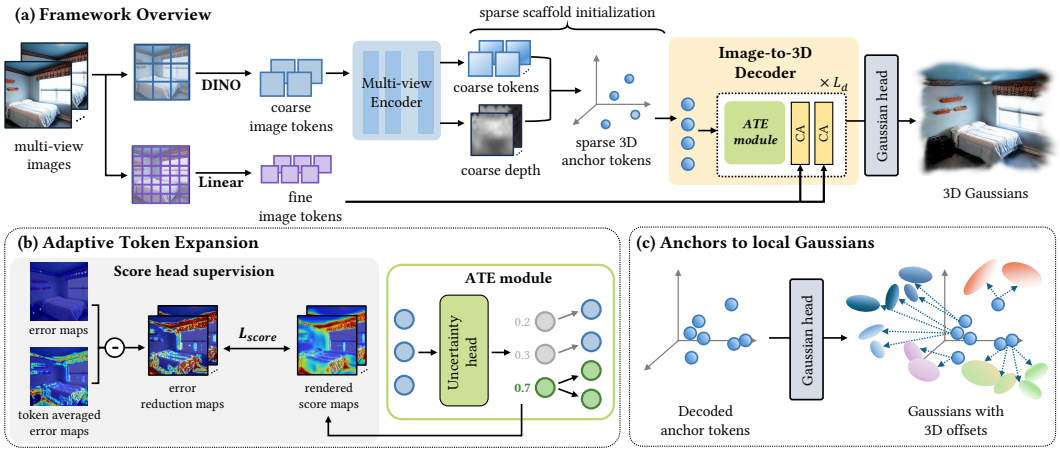


Figure 2: **(a)** Given posed multi-view images, a multi-view encoder first extracts coarse patch features and estimates patch-level depths to initialize sparse 3D anchor tokens. An image-to-3D decoder progressively expands and refines these tokens. **(b)** At each decoder block, ATE predicts a per-token expansion score and expands tokens with high scores. The score is supervised with error reduction maps, which are computed by subtracting the rendering errors of expanded tokens from those of parent-wise averaged tokens. **(c)** Each refined token is regressed into K local Gaussians with 3D offsets relative to the anchor, decoupling primitive positions from the input pixel grid.

where each Gaussian is parameterized by its center $\boldsymbol{\mu}_g$, rotation quaternion $\mathbf{q}_g$, scale $\mathbf{s}_g$, opacity α_g, and spherical harmonics $\mathbf{SH}_g$.

Many feed-forward approaches regress Gaussians in a pixel-aligned manner, where each primitive center is determined by a predicted depth and the corresponding camera ray:

$$\boldsymbol{\mu}_v(\mathbf{x}) \;=\; \mathbf{o}_v + d_v(\mathbf{x})\,\mathbf{r}_v(\mathbf{x}), \tag{2}$$

where $\mathbf{o}_v \in \mathbb{R}^3$ and $\mathbf{r}_v(\mathbf{x}) \in \mathbb{R}^3$ are the camera origin and ray direction at pixel $\mathbf{x}$. This simplifies center prediction to per-pixel depth estimation, but ties each primitive to an input pixel: the total Gaussian budget and its spatial distribution are determined by input pixel grids rather than the underlying scene. As a result, the pixel-aligned formulation produces many redundant primitives in trivial regions while under-allocating capacity to challenging regions. We therefore treat this inefficiency as a formulation-level problem: a feed-forward 3DGS model should allocate representations according to the scene's reconstruction difficulty rather than input pixel grids.

3.2 ATSplat Framework

We introduce ATSplat, a feed-forward 3DGS framework that restores the scene-adaptive capacity allocation of per-scene 3DGS optimization. Our key idea is to recast the sparse-to-adaptive principle of 3DGS as feed-forward operations centered on a set of 3D anchor tokens.

Fig. 2 (a) illustrates an overview of the framework. A multi-view image encoder first extracts coarse patch features and predicts patch-level depths. The depths are unprojected with the camera rays, assigning coarse patch features to sparse 3D locations that form a sparse anchor scaffold. An image-to-3D decoder then refines these anchor tokens by cross-attending to fine-grained image features. Within the decoder, an Adaptive Token Expansion (ATE) module selectively expands tokens based on the predicted improvement from expansion, progressively increasing capacity only where needed. Finally, each refined token is decoded into a set of local Gaussians whose centers are predicted as 3D offsets from its anchor, decoupling primitive placement from the input pixel grid.

Multi-view image encoder. The encoder extracts cross-view patch features from the input images at the coarse resolution. Each view is first tokenized into coarse patches using a DINO backbone. We also add Plücker raymap embeddings to the patch tokens to inject camera geometry. A multi-view transformer then flattens the patch tokens across all views and applies global self-attention to produce cross-view image features.

Initializing sparse anchor tokens. We then initialize a sparse 3D scaffold by computing a 3D coordinate for each encoded patch, which serve as initial anchor tokens. For each patch feature $\mathbf{f}_i \in \mathbb{R}^C$ at pixel $\mathbf{x}_i$ in view v_i, we predict a patch-level depth $\hat{d}_i$ from $\mathbf{f}_i$ with a lightweight MLP and unproject it along the corresponding ray:

$$\mathbf{p}_i \;=\; \mathbf{o}_{v_i} + \hat{d}_i\, \mathbf{r}_{v_i}(\mathbf{x}_i). \tag{3}$$

The pair $(\mathbf{p}_i, \mathbf{f}_i)$ defines a 3D anchor token–anchored at $\mathbf{p}_i$ and carrying the encoded feature $\mathbf{f}_i$. We further inject local 3D context by aggregating each anchor with its k-nearest neighbors (kNN) via a PointNet-style operator (Qi et al., 2017).

Image-to-3D decoder. The decoder refines both the features and positions of anchor tokens using fine-grained image information. For each view, we extract fine patch features at twice the coarse resolution using a lightweight per-view feature extractor. This adds Plücker raymap embeddings to the patches and applies two per-view self-attention layers. The decoder then applies a stack of L blocks, each composed of an ATE module followed by cross-attention layers. The ATE module selectively expands anchors associated with under-reconstructed regions, progressively increasing representational capacity where needed. Each block then refines token features through cross-attention and updates anchor positions with predicted 3D offsets. We describe details of ATE in Sec. 3.3.

Anchors to local 3D Gaussians. Each refined anchor token $\hat{f}_i$ is finally regressed into a set of K local Gaussian primitives. A lightweight Gaussian head, consists of 2-layer MLPs, maps the anchor feature to K sets of Gaussian attributes:

$$\big\{(\Delta\boldsymbol{\mu}_{i,k},\, \mathbf{q}_{i,k},\, \mathbf{s}_{i,k},\, \alpha_{i,k},\, \mathbf{SH}_{i,k})\big\}_{k=1}^{K} \;=\; \mathrm{MLP}(\hat{\mathbf{f}}_i), \tag{4}$$

and each Gaussian center is placed relative to the final refined anchor position $\hat{\mathbf{p}}_i$:

$$\boldsymbol{\mu}_{i,k} = \hat{\mathbf{p}}_i + \Delta\boldsymbol{\mu}_{i,k}, \tag{5}$$

which places primitive positions freely from the input pixel grid.

3.3 Adaptive token expansion

Unlike per-scene optimization, a feed-forward model cannot accumulate gradient statistics over multiple optimization steps, making it challenging to identify under-reconstructed regions. Our ATE module addresses this by learning a per-token expansion score to estimate the improvement from additional capacity. While reconstruction error could also provide useful guidance for token expansion, it does not necessarily reflect the improvement from additional capacity, as errors can also arise from insufficient input information. We therefore supervise these scores with the reduction in rendering error associated with token expansion as a proxy for this improvement.

At each decoder block, we estimate an expansion score for every anchor token $\mathbf{f}_i$ through a lightweight MLP g_ϕ:

$$u_i \;=\; g_\phi(\mathbf{f}_i) \;\in\; \mathbb{R}. \tag{6}$$

We select anchors whose scores exceed a threshold τ as expansion targets. Each selected feature is passed through a linear projection that produces M residual features, yielding M child tokens that inherit the parent's current anchor position:

$$[\Delta\mathbf{f}_{i,1},\, \ldots,\, \Delta\mathbf{f}_{i,M}] \;=\; \mathbf{W}\,\mathbf{f}_i, \qquad \mathbf{f}_{i,m} \;=\; \mathbf{f}_i + \Delta\mathbf{f}_{i,m}. \tag{7}$$

The expanded tokens together with the unselected tokens are concatenated and passed to the cross-attention layers.

Learning expansion scores in 2D space. To align the predicted scores with the improvement from token expansion, we supervise $g_\phi^{(l)}$ using the reduction in 2D rendering error. After the l-th decoder block, we apply the Gaussian head to the current anchors to obtain intermediate Gaussians $\mathcal{G}^{(l)}$ and compute their rendering error map $\mathbf{E}^{(l)}$ against ground-truth images. We also average the expanded tokens per parent before the Gaussian head and compute the corresponding rendering error map $\mathbf{E}^{(l)}_{\mathrm{avg}}$. The expected improvement map from token expansion is computed as

$$\mathbf{U}^{(l)} \;=\; \max\Big(\mathbf{E}^{(l)}_{\mathrm{avg}} - \mathbf{E}^{(l)},\, 0\Big), \tag{8}$$

Table 1: **Quantitative results on RealEstate10K at 256×256 resolution** with 2 input views. Inference times for GS-LRM and LongLRM are omitted due to the absence of public code for this setting.

Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓	Time(s)↓
PixelSplat (Charatan et al., 2024)	25.89	0.858	0.142	131K	0.127
MVSplat (Chen et al., 2024a)	26.39	0.869	0.128	131K	0.044
DepthSplat (Xu et al., 2025)	27.47	0.889	0.114	131K	0.067
GS-LRM (Zhang et al., 2024a)	28.10	0.892	0.114	131K	–
LongLRM (Chen et al., 2025)	28.54	0.895	0.109	131K	–
iLRM (Kang et al., 2026a)	28.65	0.900	0.110	131K	**0.025**
TokenGS (Ren et al., 2026)	28.41	0.903	0.135	262K	0.066
ReSplat-init (Xu et al., 2026)	<u>29.40</u>	**0.907**	**0.104**	131K	0.053
Ours ($\tau = 0.1$)	29.23	<u>0.904</u>	0.111	**15K**	<u>0.027</u>
Ours ($\tau = 0.01$)	**29.41**	**0.907**	<u>0.105</u>	<u>28K</u>	0.028

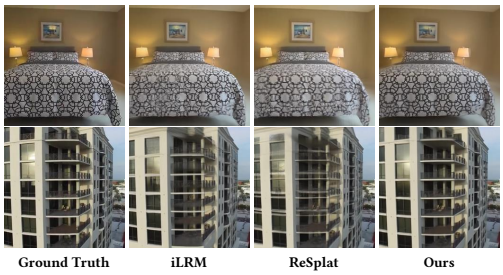


Figure 3: **Qualitative results on RealEstate10K** at 256×256 resolution with 2 input views.

where max denotes the element-wise maximum. The predicted token scores are attached to the corresponding Gaussians as scalar attributes and rasterized via standard alpha-compositing, producing a 2D score map $\hat{\mathbf{U}}^{(l)}$. We supervise $\hat{\mathbf{U}}^{(l)}$ against $\mathbf{U}^{(l)}$:

$$\mathcal{L}_{\text{score}}^{(l)} \;=\; \big\| \hat{\mathbf{U}}^{(l)} - \mathrm{sg}\big(\mathbf{U}^{(l)}\big) \big\|_2^2, \tag{9}$$

where $\mathrm{sg}(\cdot)$ stops gradients. We empirically use D-SSIM to compute these intermediate error maps.

3.4 Training

We train ATSplat end-to-end with a final rendering loss, intermediate rendering losses, and score supervision. The final rendering loss combines MSE with a perceptual term:

$$\mathcal{L}_{\text{render}} \;=\; \mathcal{L}_{\text{MSE}} + \lambda_{\text{p}} \cdot \mathcal{L}_{\text{perceptual}}. \tag{10}$$

To train the score heads at intermediate decoder blocks, we additionally supervise each intermediate Gaussian set $\mathcal{G}^{(l)}$ with an auxiliary rendering loss $\mathcal{L}_{\text{interm}}^{(l)}$. For efficiency, we replace the perceptual term with $\mathcal{L}_{\text{D-SSIM}}$ in the intermediate rendering loss. The full objective is

$$\mathcal{L} \;=\; \mathcal{L}_{\text{render}} \;+\; \lambda_{\text{smooth}} \cdot \mathcal{L}_{\text{smooth}} \;+\; \sum_{l=1}^{L} \big(\lambda_{\text{int}} \cdot \mathcal{L}_{\text{interm}}^{(l)} \;+\; \lambda_{\text{score}} \cdot \mathcal{L}_{\text{score}}^{(l)} \big), \tag{11}$$

where $\mathcal{L}_{\text{smooth}}$ is a depth smoothness loss applied to the predicted coarse depths (Xu et al., 2026).

4 Experiments

4.1 Experimental setup

Datasets. We train and evaluate ATSplat on two widely used datasets for feed-forward novel-view synthesis: RealEstate10K (Zhou et al., 2018) and DL3DV (Ling et al., 2024). RealEstate10K consists of home-tour videos collected from YouTube and mainly contains bounded indoor scenes. DL3DV provides captures of indoor and outdoor environments, covering larger spatial extents, more diverse scene layouts and complex geometry. Following previous works (Charatan et al., 2024; Chen et al., 2024a; Xu et al., 2025), we use the same preprocessing and train/test splits.

Evaluation protocols. We report PSNR, SSIM, and LPIPS (Zhang et al., 2018) for rendering quality, and measure efficiency by the number of Gaussians and inference runtime measured on a single H200 GPU. For RealEstate10K, we follow the evaluation protocol of pixelSplat (Charatan et al., 2024). For DL3DV, we follow the evaluation protocols of DepthSplat (Xu et al., 2025) at 256×448 and 512×960 resolutions, and LongLRM (Chen et al., 2025) at 540×960 resolution.

Baselines. We compare ATSplat with recent feed-forward 3DGS methods (Xu et al., 2025; Kang et al., 2026a; Xu et al., 2026; Kang et al., 2026b). For ReSplat (Xu et al., 2026), we report its initial reconstruction output (iteration 0), since its post-hoc refinement is orthogonal to the initializations and improves other feed-forward methods, as shown in the original work. For quantitative comparisons, we use reported results when methods follow the same evaluation protocol and otherwise evaluate public checkpoints under the same setup. At 512×960 resolution, we also compare with optimization-based 3DGS (Kerbl et al., 2023) and Mip-Splatting (Yu et al., 2024a), initialized with SfM (Schonberger & Frahm, 2016) points from the input views and optimized for 30K iterations.

Table 2: **Quantitative comparisons on DL3DV at 256×448 resolution.** *: reported only at 6 views, as both code and pretrained weights are not publicly available for the other setups. iLRM and ReSplat predicts pixel-aligned Gaussians at $2\times$ and $4\times$ downsampled resolution, respectively.

	2 input views				4 input views				6 input views			
Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓	PSNR↑	SSIM↑	LPIPS↓	#GS↓	PSNR↑	SSIM↑	LPIPS↓	#GS↓
MVSplat (Chen et al., 2024a)	17.54	0.529	0.402	229K	21.63	0.721	0.233	458K	22.93	0.775	0.193	688K
DepthSplat (Xu et al., 2025)	19.31	0.615	0.310	229K	23.12	0.780	0.178	458K	24.19	0.823	0.147	688K
iLRM* (Kang et al., 2026a)	–	–	–	–	–	–	–	–	25.60	0.830	0.168	172K
TokenGS (Ren et al., 2026)	19.35	0.609	0.440	262K	23.02	0.747	0.326	262K	23.69	0.766	0.311	262K
SparseSplat* (Zhang et al., 2026b)	–	–	–	–	–	–	–	–	24.20	0.817	0.168	150K
ReSplat-init (Xu et al., 2026)	18.82	0.588	0.393	14K	23.94	0.785	0.209	28K	25.78	0.838	0.162	43K
Ours ($\tau = 0.2$)	21.97	0.699	0.306	**10K**	25.87	0.821	0.191	**21K**	27.45	0.860	0.154	**32K**
Ours ($\tau = 0.1$)	**22.54**	**0.725**	**0.268**	21K	**26.81**	**0.850**	**0.151**	45K	**28.53**	**0.888**	**0.119**	68K

Implementation details. ATSplat uses a multi-view encoder with 8 global self-attention layers and PRoPE (Li et al., 2025), followed by a decoder with 3 blocks and 2 cross-attention layers per block. The DINOv2-B (Oquab et al., 2024) backbone extracts features with a patch size of 14. The coarse encoder outputs features with a patch size of 16, while the fine branch uses a patch size of 8. The fine branch uses 512-dimensional features, and the remaining components use 768-dimensional features. ATE is applied before every decoder block, selecting a fixed ratio of 0.5 of the tokens during training with an expansion ratio of 2. The score supervision is applied only to pixels covered by expanded tokens, leaving unselected tokens receive without score supervision. At evaluation, we use $\tau = 0.1$ by default and report results at additional thresholds to show the trade-off between rendering quality and Gaussian count. Each final token is decoded into $K = 8$ local Gaussians. We use $\lambda_{\text{perceptual}} = 0.5$, $\lambda_{\text{smooth}} = 0.001$, $\lambda_{\text{interm}} = 0.5$, and $\lambda_{\text{score}} = 0.1$ in all experiments. Please refer to the appendix for more details. Our code and trained weights will be made publicly available.

4.2 Results

We first evaluate ATSplat on common benchmark setups adopted in prior work (Charatan et al., 2024; Chen et al., 2024a; Xu et al., 2025). These include comparisons on RealEstate10K with two input views at 256×256 resolution, and comparisons on DL3DV at 256×448 resolution with various input view settings. Please refer to the supplements for additional video results and comparisons.

RealEstate10K. Tab. 1 reports two-view results on RealEstate10K. ATSplat achieves rendering quality comparable to ReSplat and higher PSNR and SSIM than all other baselines while using only 15K Gaussians at $\tau = 0.1$, corresponding to an $8.9\times$ reduction compared to dense pixel-aligned baselines. Fig. 3 further confirms this, showing accurate reconstruction of fine details and geometry. At $\tau = 0.01$, ATSplat reaches the rendering quality of ReSplat while still using $4.7\times$ fewer Gaussians, highlighting the effectiveness of our scene-adaptive capacity allocation. We further provide evaluations with varying viewpoint overlaps and cross-dataset generalization in the appendix.

DL3DV We next evaluate ATSplat on DL3DV, which contains more challenging scenes with complex geometry. Tab. 2 reports results at 256×448 resolution using 2, 4, and 6 input views. Across all input-view settings, ATSplat at $\tau = 0.2$ outperforms dense feed-forward baselines in PSNR and SSIM while using substantially fewer Gaussians, demonstrating that the sparse-to-adaptive formulation remains effective in larger and more diverse scenes. Reducing τ to 0.1 further improves rendering quality, achieving state-of-the-art results across all three metrics. iLRM (Kang et al., 2026a) and ReSplat (Xu et al., 2026) reduce the number of Gaussians through uniform spatial downsampling. However, this uniform reduction decreases Gaussian density across the scene, under-allocating capacity to regions with fine structures. In contrast, ATSplat progressively expands and refines selected tokens in under-reconstructed regions, achieving high-quality results with a compact Gaussian set.

4.3 High-resolution novel-view synthesis

To assess the scalability of our sparse-to-adaptive design for high-resolution rendering, we further evaluate ATSplat on DL3DV at 512×960 and 540×960 resolutions. For efficiency, we apply an additional $2\times$ downsampling to the coarse-branch input images, yielding a patch grid $4\times$ coarser than the fine grid along each spatial dimension. At 512×960, we evaluate DepthSplat (Xu et al., 2025) using its public checkpoint trained at 448×768 resolution, which is the only released configuration applicable to this setting. We also compare with optimization-based 3DGS baselines for this setup. At 540×960, we compare with methods evaluated under the 16-view LongLRM protocol.

Table 3: **Quantitative comparisons on DL3DV at** 512×960 **resolution** with 100-frame baselines. The top group lists optimization-based methods, and the bottom group lists feed-forward 3DGS methods.

Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓	Time(s)↓
3DGS (Kerbl et al., 2023)	22.92	0.760	0.227	557K	>10 m
Mip-Splatting (Yu et al., 2024a)	22.62	0.746	0.239	681K	>10 m
DepthSplat (Xu et al., 2025)	21.33	0.740	0.270	5898K	0.758 s
iLRM (Kang et al., 2026a)	24.35	0.781	0.256	1474K	0.700 s
ReSplat-init (Xu et al., 2026)	25.23	0.820	0.205	368K	0.421 s
Ours ($\tau = 0.1$)	25.75	0.806	0.244	**121K**	**0.126 s**
Ours ($\tau = 0.01$)	**26.42**	**0.836**	**0.200**	286K	0.249 s

Table 4: **Quantitative comparisons on DL3DV at** 540×960 **resolution** with 16 input views. Models are evaluated under the LongLRM protocol. ReSplat is reported without undistortion as in the original work.

Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓	Time(s)↓
LongLRM (Chen et al., 2025)	22.66	0.740	0.292	8356K	0.260
iLRM (Kang et al., 2026a)	22.91	0.766	0.295	2089K	0.268
MVP (Kang et al., 2026b)	23.76	0.798	0.239	8356K	**0.091**
tttLRM (Wang et al., 2026a)	23.60	0.784	0.255	8232K	1.572
ReSplat-init (Xu et al., 2026)	22.69	0.742	0.307	518K	0.761
Ours ($\tau = 0.1$)	24.01	0.790	0.279	**137K**	0.243
Ours ($\tau = 0.01$)	**24.66**	**0.820**	**0.234**	358K	0.405



Figure 4: **Qualitative comparisons on DL3DV at** 540×960 **resolution** with 16 input views. ATSplat produces high-fidelity renderings with only 358K Gaussians, which is up to $23\times$ fewer than dense pixel-aligned 3DGS methods. It also faithfully reconstructs thin structures and details.

As shown in Tab. 3, ATSplat reconstructs high-resolution scenes in a single forward pass and completes inference in under 0.25 seconds, achieving high-quality rendering with far fewer Gaussians than pixel-aligned feed-forward methods. For the 16-view 540×960 setting in Tab. 4, ATSplat at $\tau = 0.1$ achieves competitive rendering quality while using $61\times$ fewer Gaussians than MVP (Kang et al., 2026b). At $\tau = 0.01$, ATSplat outperforms MVP across all three quality metrics while still using $23.3\times$ fewer Gaussians. Fig. 4 further shows that ATSplat preserves fine structures more faithfully than compact feed-forward baselines, supporting the advantage of our progressive, scene-adaptive allocation over uniform spatial downsampling.

4.4 Ablations and Analysis

To examine the effects of our core components, we conduct ablations and analyses on RealEstate10K with two input views and $\tau = 0.1$ by default. Please refer to the appendix for additional analysis.

3D anchor designs. We compare our anchor-offset formulation with three alternatives. The first is a pixel-aligned ray-depth formulation, where Gaussians are tied to input pixels and lifted along camera rays. The second uses learnable initial tokens without coarse scene-specific geometry. The third directly regresses absolute 3D Gaussian centers instead of local offsets around coarse anchors.

As shown in Tab. 5, the proposed anchor-offset design achieves the best reconstruction quality. The pixel-aligned variant constrains Gaussians to input pixels, which limits the flexibility of our adaptive expansion. The latter two variants lack scene-specific geometric priors, leaving Gaussian placement to be learned without spatial guidance. These results indicate that coarse 3D anchors provide meaningful initialization, while local offsets allow expanded tokens to place Gaussians freely in 3D.

Selection strategy. We further ablate the selection strategy in ATE. All variants are matched to use approximately 13–15K Gaussians. As shown in Tab. 6, random selection and FPS provide limited gains as they allocate capacity without targeting difficult regions. We also compare with a fully learnable variant using a straight-through estimator (STE) (Yin et al., 2019), following a

Table 5: **Ablation results on the 3D anchor designs.**

Anchor init.	Gaussian center	PSNR↑	SSIM↑	LPIPS↓
None	Ray + depth	27.98	0.884	0.132
Learnable	Direct xyz	<u>28.02</u>	<u>0.885</u>	<u>0.125</u>
None	Direct xyz	<u>28.02</u>	0.882	0.139
Patch depth	Anchor + offset	**29.23**	**0.904**	**0.111**

Table 6: **Ablation results on the ATE module.**

Method	PSNR↑	SSIM↑	LPIPS↓
Random	28.70	0.897	<u>0.112</u>
Farthest Point Sampling (FPS)	28.65	0.895	0.120
Fully learnable (STE) (Yin et al., 2019)	28.77	0.895	<u>0.112</u>
Rendering error	<u>29.03</u>	<u>0.903</u>	<u>0.112</u>
Error reduction	**29.23**	**0.904**	**0.111**

Table 7: **Ablation results on token expansion**. Single-stage variants expand tokens at the beginning or end of the stage using $M = 8$, matching the maximum expansion factor of progressive expansion with $M = 2$ at each of three stages.

Method	PSNR↑	SSIM↑	LPIPS↓
No expansion	27.34	0.867	0.160
Single-stage (expand first)	<u>28.48</u>	<u>0.891</u>	<u>0.124</u>
Single-stage (expand last)	28.43	0.890	0.127
Progressive expansion	**29.23**	**0.904**	**0.111**

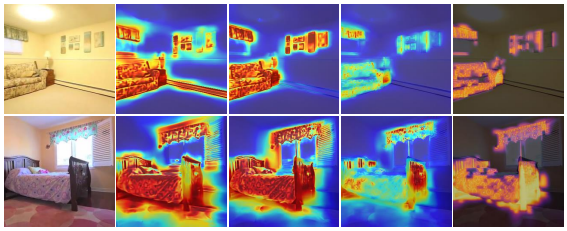


Figure 5: **Visualization of predicted score maps**. We visualize Gaussian alpha maps of selected tokens at stage 3.

prior upsampling design (Nam et al., 2025), but find that it performs similarly to simple heuristic selection. While rendering-error supervision also provides useful guidance for token selection, our error reduction supervision better identifies under-reconstructed regions that are likely to benefit from expansion, leading to further improvements.

Token expansion. To validate our progressive token expansion, we compare against a variant without expansion and two single-stage variants that expand tokens at the beginning or end of the stage. Both single-stage variants use $M = 8$ to match the maximum expansion factor of our model ($M = 2$ at each of three stages), with thresholds adjusted for similar Gaussian counts (13–15K).

As shown in Tab. 7, removing token expansion substantially reduces rendering quality. Progressive expansion also outperforms both single-stage variants across all three metrics. These results highlight the effectiveness of our progressive token expansion scheme, where tokens expanded and refined at each stage guide better capacity allocation and expansion decisions at subsequent stages.

Predicted expansion scores and selected tokens. Finally, we analyze how the predicted expansion scores change across decoder stages and guide token selection. As shown in Fig. 5, early-stage scores highlight relatively broad regions around objects and their boundaries, while high scores become more localized around fine structures and textured regions in later stages. This suggests that the score head adjusts its predictions as tokens are expanded and refined across decoder stages. At stage 3, the selected tokens concentrate in these high-score regions, illustrating how the predicted expansion scores guide additional capacity toward regions expected to benefit from expansion.

5 Conclusion and Future Work

We presented ATSplat, a feed-forward 3D Gaussian Splatting framework that restores the scene-adaptive capacity allocation of the 3DGS pipeline. Instead of assigning Gaussians to input image grids, ATSplat initializes sparse 3D anchor tokens and progressively selects, expands, and refines them in under-reconstructed regions. Each refined token is then decoded into a set of local Gaussians with 3D relative offsets, allowing flexible Gaussian placement in 3D. This sparse-to-adaptive formulation allocates capacity according to scene complexity. ATSplat reinterprets the core design principle of 3DGS within a feed-forward framework, achieving competitive rendering quality while using up to $61\times$ fewer Gaussians than dense pixel-aligned 3DGS methods.

Limitations and future work. While ATSplat brings progressive, scene-adaptive capacity allocation to feed-forward 3DGS, several directions remain open for future work. First, ATSplat adaptively expands tokens but does not prune them. While opacity-based pruning (Chen et al., 2025) can also be applied to our final Gaussians, pruning redundant tokens during decoding could further improve decoder efficiency. Second, replacing the multi-view encoder with 3D foundation models (Wang et al., 2025a) and extending ATSplat to unposed settings would broaden its applicability.

AI USE STATEMENT

We used generative AI tools to improve the wording and flow of the manuscript and to assist with coding. All AI-assisted text and code were reviewed and verified by the authors, who take full responsibility for the final content and results.

ETHICS STATEMENT

This work studies 3D reconstruction and novel-view synthesis using existing datasets. Applications of this technology should respect the privacy of captured spaces and clearly distinguish synthesized views from real observations.

REPRODUCIBILITY STATEMENT

We describe our method and training objectives in Section 3 and provide experimental settings and evaluation protocols in Section 4.1. Appendix A provides additional training and architecture details. Our code and trained weights will be made publicly available.

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 (CVPR)*, pp. 363–373, 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)*, pp. 19457–19467, 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.

Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. MVSplat360: Feed-forward 360 scene synthesis from sparse views. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 107064–107086, 2024b.

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

Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned Gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5354–5363, 2024.

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 (ICML)*, pp. 23662–23681, 2025.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conference Papers (SIGGRAPH)*, pp. 1–11, 2024.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 171–189, 2026.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, Dahua Lin, and Bo Dai. AnySplat: Feed-forward 3D Gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 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)*, pp. 37332–37342, 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)*, pp. 37380–37390, 2026b.

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

Injae Kim, Chaehyeon Kim, Minseong Bae, Minseok Joo, and Hyunwoo J. Kim. F^4Splat: Feed-forward predictive densification for feed-forward 3D Gaussian splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 1–19, 2026.

Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 15984–16009, 2025.

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

Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite Nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14458–14467, 2021.

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

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 (CVPR)*, pp. 20654–20664, 2024.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

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)*, pp. 11756–11766, 2026.

Seungtae Nam, Xiangyu Sun, Gyeongjin Kang, Younggeun Lee, Seungjun Oh, and Eunbyung Park. Generative densification: Learning to densify Gaussians for high-fidelity generalizable 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26683–26693, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, 2017.

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)*, pp. 15365–15375, 2026.

Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4104–4113, 2016.

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

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 36582–36592, 2026a.

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)*, pp. 5294–5306, 2025a.

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)*, pp. 1–19, 2026b.

Yiming Wang, Lucy Chai, Xuan Luo, Michael Niemeyer, Manuel Lagunas, Stephen Lombardi, Siyu Tang, and Tiancheng Sun. Learning efficient fuse-and-refine for feed-forward 3D Gaussian splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 76311–76341, 2025b.

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 (NeurIPS)*, volume 37, pp. 107326–107349, 2024.

Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. FreeSplat++: Generalizable 3D Gaussian splatting for efficient indoor scene reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 48(7):7749–7765, 2026c.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20310–20320, 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)*, pp. 16453–16463, 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)*, pp. 364–382, 2026.

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4D Gaussian splatting. In *International Conference on Learning Representations (ICLR)*, volume 2024, pp. 9142–9159, 2024.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3D Gaussian splats from sparse unposed images. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 54009–54033, 2025.

Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. YoNoSplat: You only need one model for feedforward 3D Gaussian splatting. In *International Conference on Learning Representations (ICLR)*, pp. 39852–39871, 2026.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6):1–13, 2024b.

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)*, 2024a.

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 (CVPR)*, pp. 586–595, 2018.

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 (NeurIPS)*, volume 37, pp. 50361–50380, 2024b.

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, 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)*, pp. 5049–5058, 2026b.

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

Table 8: **Quantitative comparisons on RealEstate10K** with varying viewpoint overlaps. Smaller viewpoint overlaps indicate larger camera baseline ranges.

		Small			Medium			Large			Average		
Method	#GS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
pixelSplat (Charatan et al., 2024)	131K	20.28	0.719	0.265	23.73	0.811	0.180	27.15	0.880	0.121	23.86	0.808	0.184
MVSplat (Chen et al., 2024a)	131K	20.37	0.725	0.250	23.81	0.814	0.172	27.47	0.885	0.115	24.01	0.812	0.175
DepthSplat (Xu et al., 2025)	131K	22.82	0.798	0.193	25.38	0.851	0.145	28.32	0.900	0.104	25.59	0.852	0.145
iLRM (Kang et al., 2026a)	131K	23.82	0.813	0.184	26.54	0.864	0.139	29.43	0.910	0.103	26.70	0.864	0.140
ReSplat-init (Xu et al., 2026)	131K	24.38	<u>0.822</u>	<u>0.179</u>	<u>27.20</u>	0.871	<u>0.134</u>	**30.24**	**0.915**	**0.097**	<u>27.38</u>	<u>0.871</u>	<u>0.135</u>
Ours ($\tau = 0.1$)	**14K**	<u>24.44</u>	<u>0.822</u>	0.186	27.18	<u>0.871</u>	0.139	30.04	<u>0.912</u>	0.103	27.33	<u>0.871</u>	0.140
Ours ($\tau = 0.01$)	<u>27K</u>	**24.59**	**0.828**	**0.174**	**27.35**	**0.875**	**0.131**	<u>30.22</u>	**0.915**	<u>0.098</u>	**27.49**	**0.875**	**0.132**

A Additional implementation details

In this section, we provide additional details on the model architecture and training configuration used in our experiments. These implementation details are included to facilitate reproducibility.

A.1 Training details

We train our models with the PyTorch framework. To improve memory efficiency during training, we employ the native scaled dot-product attention implementation provided by PyTorch and train the model using BF16 precision. We train the model using AdamW (Loshchilov & Hutter, 2019) with a weight decay of 0.05 and cosine annealing after warm-up. The DINO backbone starts at a learning rate of 2×10^{-6}, and the remaining parameters start at 2×10^{-4}. ATSplat is first trained on RealEstate10K with 2 input views using 4 RTX 4090 GPUs for about two days, with a batch size of 8 per GPU, and serves as the base model. For the DL3DV experiments at 256×448 with 2–6 input views, we train the model on 4 H200 GPUs for about two days with a batch size of 8 per GPU. For the high-resolution setting, we fine-tune this model with 8–12 input views using 8 H200 GPUs with a batch size of 2 per GPU for about two days. To evaluate under the LongLRM protocol, the model is further fine-tuned on undistorted images with 16 input views using 8 H200 GPUs with a batch size of 2 per GPU for about two days.

A.2 Architecture details

The DINO backbone uses a patch size of 14, while the coarse encoder and fine branch use patch sizes of 16 and 8, respectively. Each decoder block consists of two standard cross-attention layers. Each of these layers consists of cross-attention, self-attention, and a feed-forward network.

In addition to DINO features, we patchify the raw images using a patch size of 16 and fuse the resulting image patch features with the corresponding DINO features through concatenation and a linear projection layer, after projecting them to the model dimension. For the fine branch, before feeding the fine-resolution patch features to the decoder, we also inject the encoder output features into them as coarse-level context. The final coarse patch features are upsampled to the spatial resolution of the fine patch grid and added to the fine patch features. At each decoder block, we also update the fine patch features by rendering the intermediate Gaussians at the input views and fusing the rendered outputs into the patch features, which further improves token refinement.

The coarse depth head and the Gaussian head are both implemented using two-layer MLPs. Before each MLP head, we apply LayerNorm to stabilize the feature distribution. The coarse depth head uses $\mathrm{ELU}(\cdot) + 1 + d_{\mathrm{near}}$ as its output activation, where d_{near} denotes the near-plane depth. For the uncertainty head, we apply a softplus activation to the output. We empirically found this parameterization to be more stable in our setup. To normalize the input camera poses, we compute the average camera pose and transform the camera coordinate system such that the average pose is aligned with the identity pose, following prior work (Kang et al., 2026a;b).

B Additional results

B.1 Results with varying viewpoint overlaps

We further evaluate ATSplat under varying input viewpoint overlaps, following the evaluation protocol of (Ye et al., 2025). Tab. 8 reports PSNR, SSIM, and LPIPS on RealEstate10K across the

Table 9: **Quantitative comparisons on DL3DV at 224×224 resolution** with 6 input views. Input and target images are center-cropped and resized following the YoNoSplat evaluation protocol.

Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓
YoNoSplat (Ye et al., 2026)	<u>24.74</u>	<u>0.817</u>	<u>0.139</u>	<u>301K</u>
Ours ($\tau = 0.1$)	**27.91**	**0.877**	**0.118**	**31K**

Table 10: **Extrapolation results on DL3DV** with 6 input views. Targets are the 10 frames immediately before and 10 after the input window.

Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓
DepthSplat (Xu et al., 2025)	19.26	0.689	<u>0.264</u>	688K
ReSplat-init (Xu et al., 2026)	19.48	0.708	0.273	<u>43K</u>
Ours ($\tau = 0.2$)	<u>21.78</u>	<u>0.743</u>	0.265	**32K**
Ours ($\tau = 0.1$)	**22.48**	**0.767**	**0.233**	67K

Table 11: **Cross-dataset evaluation on DL3DV** at 256×256 with 2 input views. Models trained on RealEstate10K are evaluated without fine-tuning. ReSplat is omitted since its depth model was pretrained on both RealEstate10K and DL3DV.

Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓
MVSplat (Chen et al., 2024a)	16.23	0.458	0.440	131K
DepthSplat (Xu et al., 2025)	18.07	0.568	**0.352**	131K
iLRM (Kang et al., 2026a)	18.82	<u>0.584</u>	**0.352**	131K
Ours ($\tau = 0.1$)	<u>18.96</u>	0.572	0.391	**9K**
Ours ($\tau = 0.01$)	**19.29**	**0.587**	<u>0.355</u>	<u>27K</u>

Table 12: **Cross-dataset evaluation on ACID** at 256×256 with 2 input views. Models trained on RealEstate10K are evaluated without fine-tuning.

Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓
PixelSplat (Charatan et al., 2024)	27.64	0.830	0.160	131K
MVSplat (Chen et al., 2024a)	28.15	0.841	0.147	131K
GS-LRM (Zhang et al., 2024a)	28.84	0.849	0.146	131K
DepthSplat (Xu et al., 2025)	28.37	0.847	0.141	131K
iLRM (Kang et al., 2026a)	29.24	0.856	0.143	131K
ReSplat-init (Xu et al., 2026)	<u>29.67</u>	<u>0.863</u>	**0.137**	131K
Ours ($\tau = 0.1$)	29.65	0.860	0.148	**18K**
Ours ($\tau = 0.01$)	**29.82**	**0.864**	<u>0.138</u>	<u>29K</u>

resulting overlap bins. ATSplat at $\tau = 0.01$ outperforms ReSplat-init across all three metrics in the small- and medium-overlap settings and achieves comparable quality in the large-overlap setting, while using $4.8\times$ fewer Gaussians on average. Pixel-aligned methods, including ReSplat-init, distribute Gaussians densely over the input image grids, which naturally favors reproducing the observed views and nearby viewpoints. As the rendering viewpoint departs further from the inputs, errors in scene geometry and spatial allocation become more apparent. Our anchor-offset design allows tokens expanded in under-reconstructed regions to place Gaussians freely in 3D, giving progressive expansion the flexibility to allocate capacity beyond the input image grids.

B.2 Results with square-cropped inputs

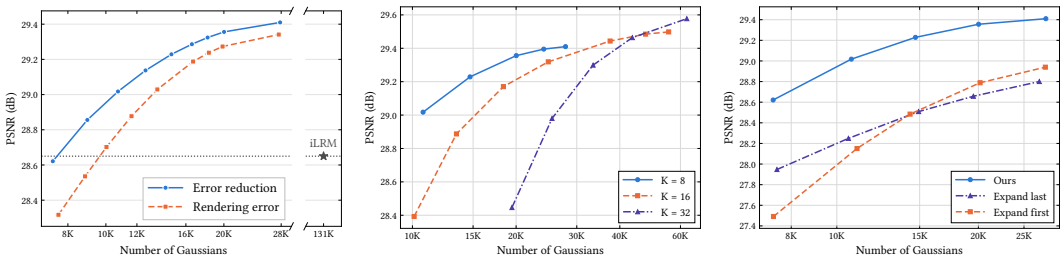
We further compare ATSplat with YoNoSplat (Ye et al., 2026), a concurrent feed-forward method evaluated on square-cropped images. To match its configuration, we train and evaluate ATSplat on DL3DV with 6 input views, using input and target images center-cropped and resized to 224×224. As shown in Tab. 9, ATSplat achieves higher quality across all metrics while using $9.7\times$ fewer Gaussians on average. YoNoSplat first predicts a dense set of Gaussians and prunes them based on opacity, while ATSplat progressively expands sparse anchor tokens where additional capacity is needed. These results highlight the effectiveness of our progressive and selective expansion, which directs additional capacity to regions where it can improve reconstruction.

B.3 Extrapolation

Extrapolated views, which lie outside the input view window, pose a harder setting than the standard protocol, where target views are placed between the input views. On DL3DV with 6 input views, we render the 10 frames immediately before and the 10 frames immediately after the input view window. As shown in Tab. 10, ATSplat at $\tau = 0.1$ achieves higher quality across all three metrics than DepthSplat (Xu et al., 2025) and ReSplat-init (Xu et al., 2026). Pixel-aligned methods tie primitive placement to input image grids, limiting their flexibility in allocating capacity across 3D space. Our 3D anchor design with local Gaussians decouples both token and Gaussian placement from the input rays, giving sparse-to-adaptive expansion the flexibility to concentrate capacity in under-reconstructed regions.

B.4 Cross-dataset generalization

We next evaluate cross-dataset generalization by testing models trained on RealEstate10K directly on DL3DV and ACID (Liu et al., 2021) without fine-tuning, using 2 input views at 256×256 resolution. For DL3DV, we use the same 50-frame sequences and input views as the 2-view setting in Tab. 2, following the DepthSplat evaluation protocol (Xu et al., 2025). Both input and target images are square-cropped to 256×256. As shown in Tab. 11, ATSplat at $\tau = 0.01$ achieves the highest



(a) (b) (c)

Figure 6: **Rendering quality across Gaussian budgets. (a) Expansion score supervision.** The PSNR advantage of error reduction over rendering error becomes larger at smaller budgets. **(b) Number of Gaussians per token.** We vary the threshold τ for models with $K = 8, 16, 32$ Gaussians per token. **(c) Expansion strategy.** Our progressive expansion consistently outperforms both single-stage variants across Gaussian budgets.

PSNR and SSIM while using only 27K Gaussians, compared with 131K for the baselines. A similar advantage is observed on ACID (Tab. 12), where ATSplat at $\tau = 0.01$ achieves the highest PSNR and SSIM with 29K Gaussians. These results suggest that the learned scoring function remains effective in guiding token expansion on unseen scenes.

C Additional Analysis and Ablations

C.1 Results with Varying Gaussians per Token

We evaluate the effect of varying the number of Gaussians decoded from each anchor token, K. We compare $K = 4, 8, 16, 32$ on RealEstate10K with 2 input views, keeping all other settings identical. As shown in Tab. 13, increasing K consistently improves rendering quality while increasing the Gaussian count. We use $K = 8$ as our default setting considering the trade-off between the Gaussian budget and rendering quality gains.

Table 13: **Results with varying numbers of Gaussians per token** K. All models use $\tau = 0.1$. We adopt $K = 8$ by default.

Method	PSNR↑	SSIM↑	LPIPS↓	#GS↓
$K = 4$	28.93	0.898	0.118	**8.2K**
$K = 8$	29.23	0.904	0.111	14.7K
$K = 16$	29.32	0.906	0.107	24.8K
$K = 32$	**29.46**	**0.907**	**0.105**	43.5K

C.2 Gaussian Budgets

We examine how rendering quality changes with the Gaussian budget by varying the selection threshold τ at inference. As shown in Fig. 6 (a), ATSplat achieves higher PSNR as more Gaussians are allocated, with smaller gains at larger budgets. Even at approximately 9K Gaussians, ATSplat achieves higher PSNR than all baselines except ReSplat-init in Tab. 1. Our error reduction supervision also consistently outperforms rendering-error supervision at similar Gaussian counts. These results suggest that error reduction better identifies regions that would benefit from additional capacity, leading to improved rendering quality particularly at smaller Gaussian budgets.

We further compare models with $K = 8, 16, 32$ Gaussians per token by varying the threshold τ. As shown in Fig. 6 (b), models with larger K reach higher PSNR at larger budgets, while $K = 8$ performs better at similar Gaussian counts. These results show that using more tokens with fewer Gaussians per token improves rendering quality under limited Gaussian budgets, at the cost of increased decoder computation.

We finally compare the two single-stage expansion variants in Tab. 7 across different Gaussian budgets. As shown in Fig. 6 (c), the expand-first variant performs particularly poorly at small Gaussian budgets. This variant predicts expansion scores before token refinement and therefore cannot use refined token features to guide expansion decisions. In contrast, the expand-last variant shows limited gains as the budget increases, which is attributed to insufficient refinement of the expanded tokens. Our progressive expansion consistently outperforms both variants across all Gaussian budgets, confirming the effectiveness of our sparse-to-adaptive expansion scheme.

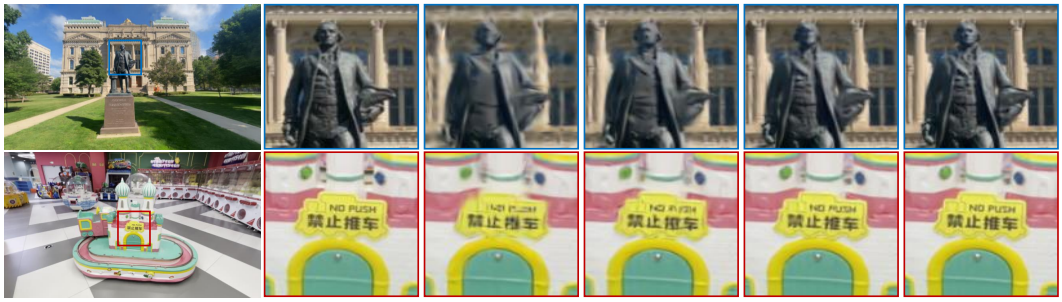


Figure 7: **Effect of progressively enabling token expansion across decoder blocks.** Starting from a configuration with expansion disabled at all decoder blocks, we gradually enable expansion from the earliest block to the latest. As more blocks are activated, fine-grained texture and thin structures are progressively recovered.

C.3 Expansion at each decoder block

To examine the effect of token expansion at each decoder block, we progressively enable expansion from the earliest block to the latest, while keeping all other settings identical. As shown in Fig. 7, the model fails to reconstruct some fine textures and thin structures without token expansion. As expansion is enabled at subsequent decoder blocks, the model progressively reconstructs finer details and more complex regions. These results illustrate how our sparse-to-adaptive design progressively allocates capacity to under-reconstructed regions throughout decoding.

D Additional qualitative results

We provide additional qualitative comparisons on RealEstate10K and DL3DV. Fig. 8 presents results on RealEstate10K. Fig. 9 and Fig. 10 present results on the DL3DV dataset at 256×448 and 540×960 resolutions, respectively. We further refer readers to the supplementary video for more comprehensive results, which include rendered sequences of ATSplat and side-by-side comparisons.

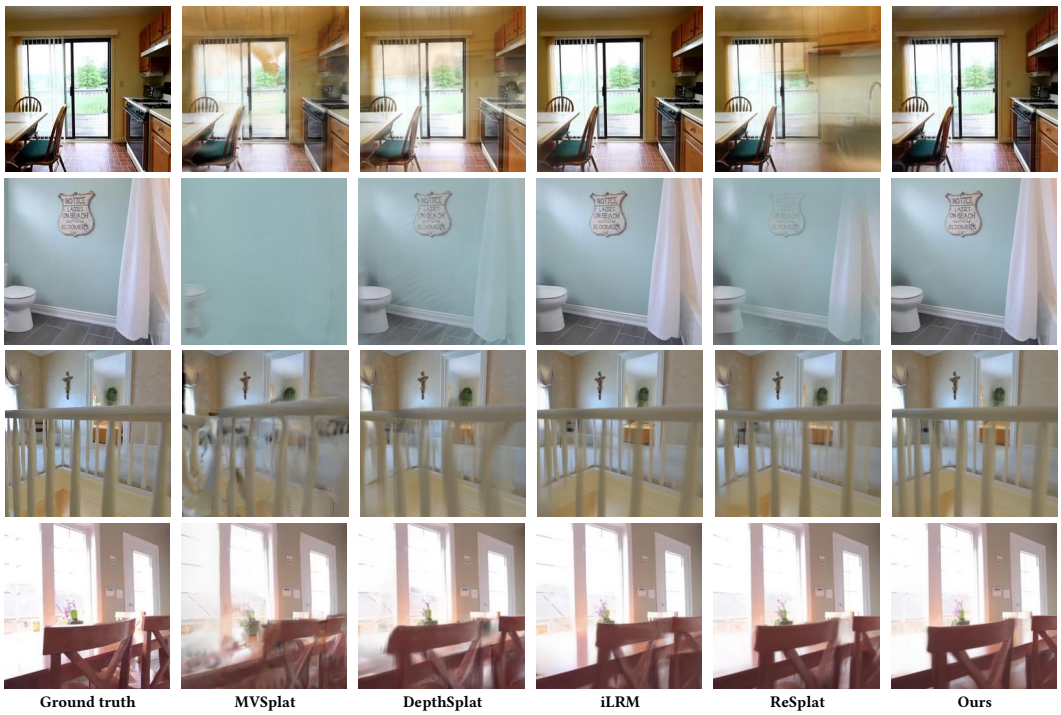


Figure 8: **Additional qualitative comparisons on RealEstate10K at** 256×256 **resolution** with 2 input views.

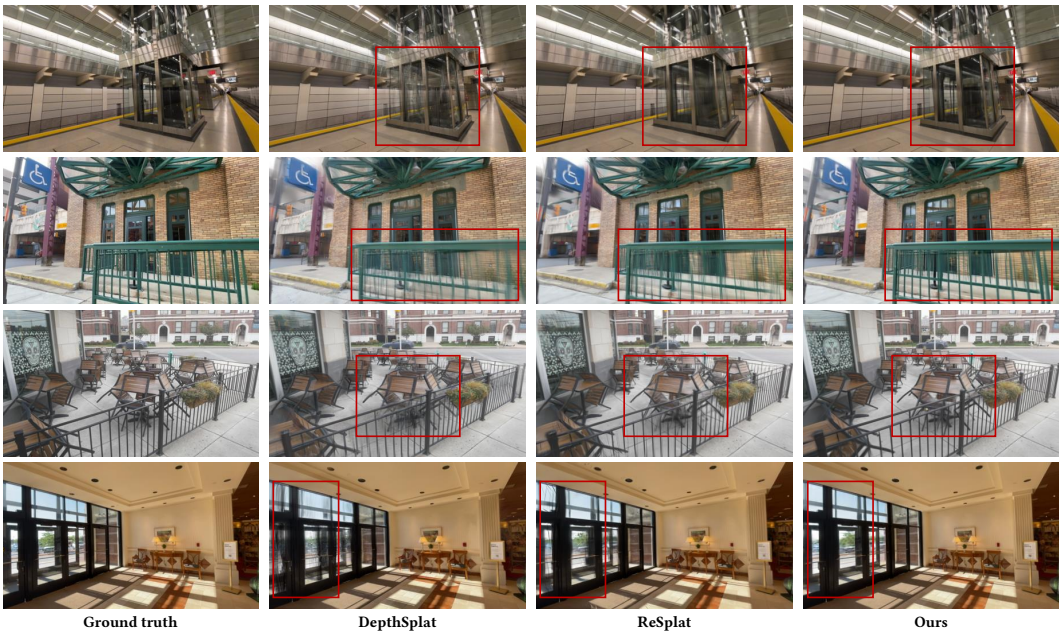


Figure 9: **Additional Qualitative comparisons on DL3DV at** 256×448 **resolution** with 6 input views.

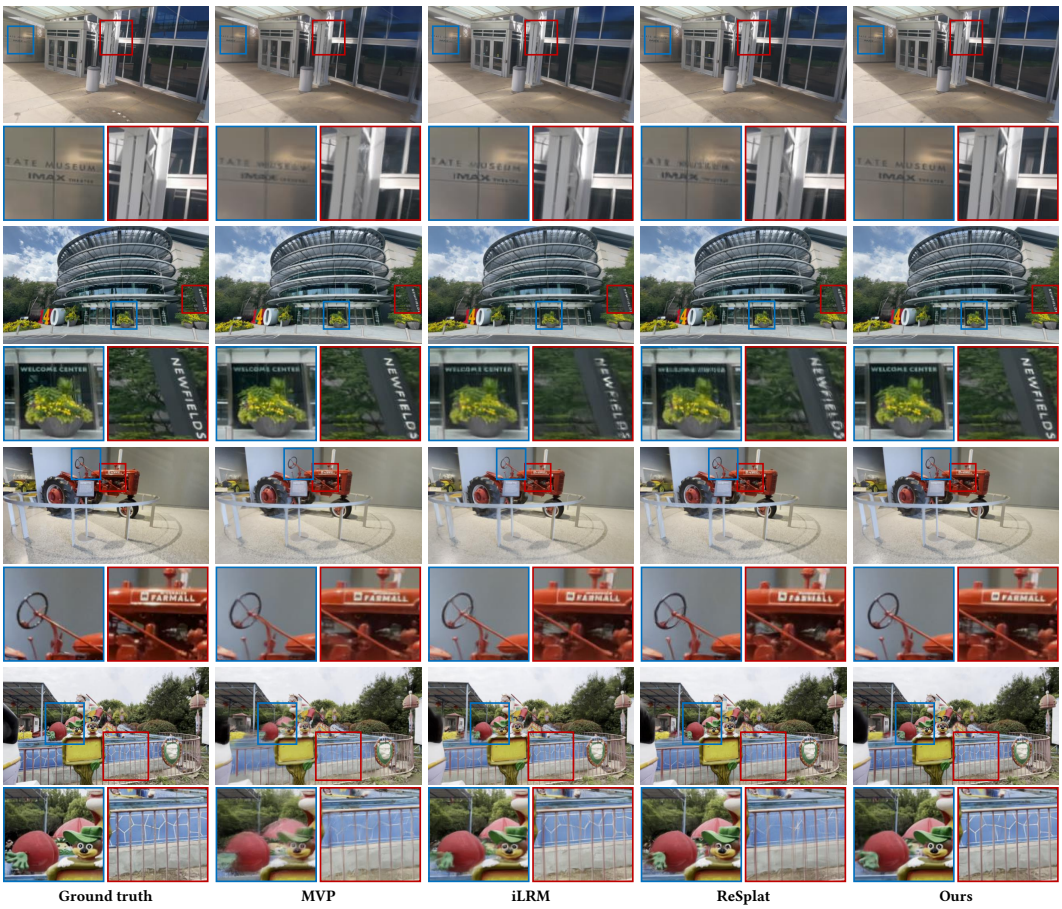


Figure 10: **Additional Qualitative comparisons on DL3DV at** 540×960 **resolution** with 16 input views.