ATSplat: Compact Feed-forward 3D Gaussian Splatting with Adaptive Token Expansion

IN CHO, Yonsei University, South Korea
JEONGHWAN CHO, Yonsei University, South Korea
MIJIN YOO, Yonsei University, South Korea
GIM HEE LEE, National University of Singapore, Singapore
SEON JOO KIM, Yonsei University, South Korea

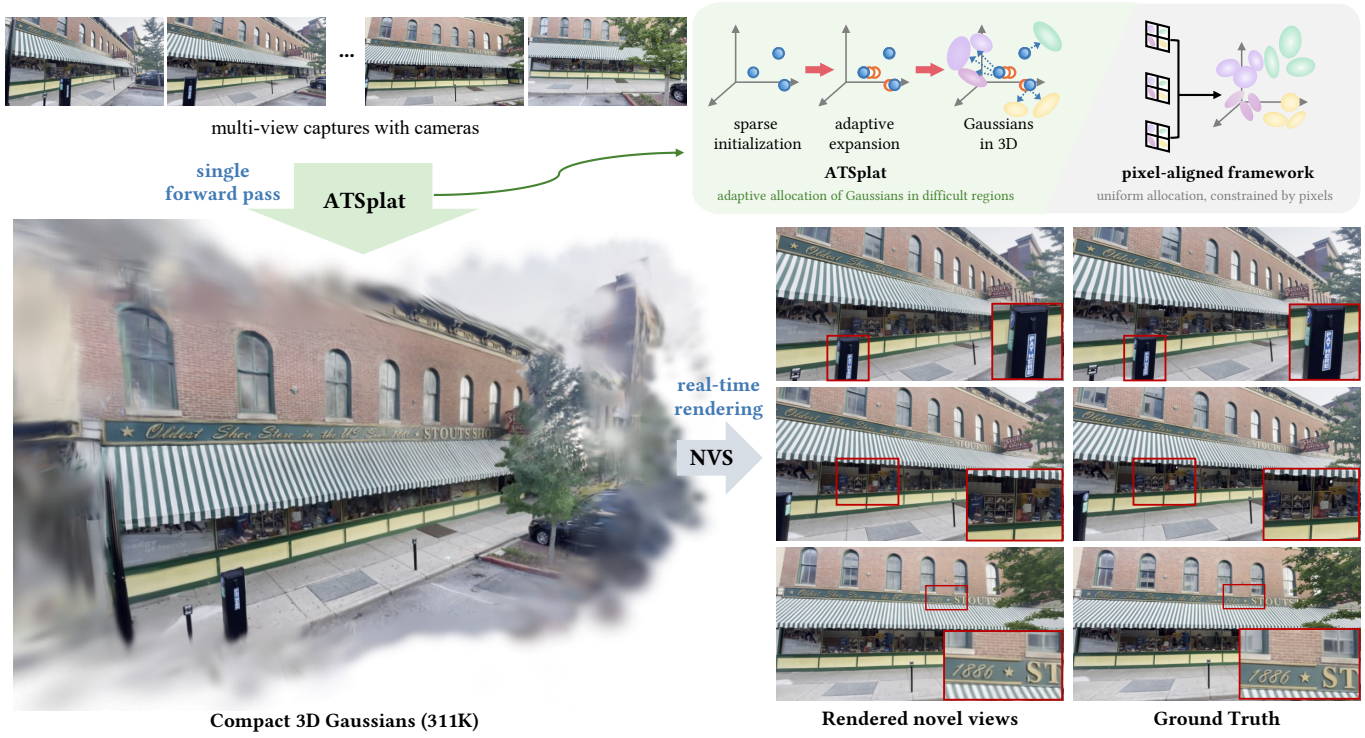

Fig. 1. ATSplat reconstructs compact 3D Gaussians from multi-view captures in a single forward pass. Unlike dense pixel-aligned feed-forward methods that place Gaussians according to input image grids, ATSplat starts from sparse 3D anchor tokens and adaptively expands tokens associated with challenging regions. This sparse-to-adaptive allocation concentrates representational capacity according to scene complexity, enabling high-quality novel-view rendering with only 311K Gaussians. With 12 images at 512 × 960 resolution, ATSplat completes reconstruction in less than a second, which can be rendered at 1136 FPS.

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, which commonly regress Gaussians at input pixels and lift them along camera rays. Such pixel-aligned formulations make the number and placement of primitives depend on image resolution and input viewpoints rather than scene complexity, resulting in dense and often redundant Gaussian sets. We present ATSplat, a feed-forward 3DGS framework that restores the adaptive allocation capability of 3DGS optimization through Adaptive 3D Tokens. ATSplat first lifts coarse patch-level depth and camera cues into sparse 3D anchor tokens, forming a compact scaffold of the scene. Each token is then regressed into local Gaussians with learnable 3D offsets, decoupling primitive placement from input image grids. An Adaptive Token Expansion module predicts a token-level uncertainty score, supervised by rendering error maps, and selectively expands high-uncertainty tokens through learnable expansion layers. This sparse-to-adaptive formulation enables ATSplat to concentrate primitives in challenging regions while maintaining a compact representation. Experiments on two representative datasets, RealEstate10K and DL3DV, show that ATSplat achieves state-of-the-art rendering quality while reducing the number of Gaussians by more than 5.7× compared with dense feed-forward 3DGS methods. From 12 input images at 512 × 960 resolution, ATSplat completes reconstruction in less than a second using a single commercial GPU, and renders high-quality novel views at 1136 FPS (512 × 960) with only 311K Gaussians. Our project page is at: https://join16.github.io/page-atsplat

Authors' Contact Information: In Cho, Yonsei University, South Korea; Jeonghwan Cho, Yonsei University, South Korea; Mijin Yoo, Yonsei University, South Korea; Gim Hee Lee, National University of Singapore, Singapore; Seon Joo Kim, Yonsei University, South Korea.

CCS Concepts: • **Computing methodologies → Rendering**; *Reconstruction*; Neural networks.

Additional Key Words and Phrases: 3D Gaussian Splatting, Multi-view capture, Novel-view synthesis, Feed-forward models

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, primitives are freely positioned throughout 3D space, and adaptive densification redirects representational capacity toward under-reconstructed regions. This scene-adaptive capacity allocation–primitives placed and refined according to the scene's complexity–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 in many applications. A growing body of work therefore seeks to bypass this cost, by leveraging learned priors to reconstruct Gaussian primitives directly from multi-view captures 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; Kang et al. 2025; Xu et al. 2025]: 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 and achieves impressive results, it also produces vast amounts of dense, per-pixel Gaussians. As the number of primitives grows with both image resolution and the number of input views, these methods often suffer from significant rendering inefficiency and storage costs.

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 depend on camera sampling rather than reconstruction difficulty. As a result, they sacrifice a key strength of optimized 3DGS: the ability to allocate capacity adaptively according to scene complexity. Closing this gap calls for a feed-forward design in which Gaussian placement reflects scene complexity instead of image structure.

We present ATSplat, a feed-forward 3DGS framework that restores scene-adaptive capacity allocation of 3DGS optimization with Adaptive Tokens. Our framework reinterprets three core design principles of per-scene 3DGS optimization–sparse initialization, free placement in 3D, and adaptive capacity allocation–as feed-forward operations centered around our adaptive anchor tokens. First, ATSplat lifts coarse patch-level depth and camera information into sparse 3D anchor tokens, forming a compact scaffold of the scene. Second, each token is decoded into a small set of local Gaussians whose positions are predicted relative to the token anchor, decoupling primitive placement from input image grids. Third, ATSplat progressively expands tokens associated with challenging regions, concentrating representational capacity where the scene demands it. Together, this sparse-to-adaptive formulation produces a compact set of Gaussians distributed according to scene complexity.

The central challenge in adaptive allocation is identifying under-reconstructed regions without access to rendering errors, as these errors are not directly observable in a single forward pass. Our Adaptive Token Expansion module addresses this by learning a per-token uncertainty score, supervised through rendered uncertainty maps to match actual rendering errors. Coupled with the anchor-based flexible 3D placement, the adaptive expansion enables us to concentrate representational capacity to challenging regions.

Thanks to the flexible and adaptive anchor design, ATSplat uses its representation budget more efficiently compared to 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 state-of-the-art rendering quality while reducing the number of Gaussians by more than 5.7×. These results suggest that high-quality feed-forward 3DGS arises from how representation capacity is allocated, not from how densely it is sampled. By restoring core design principles of 3DGS, ATSplat achieves state-of-the-art quality with more compact representation and real-time rendering.

Our key contributions can be summarized as follows:

- We propose ATSplat, a feed-forward 3DGS framework built around adaptive 3D anchor tokens that restores the scene-adaptive capacity allocation of 3DGS in feed-forward setups.
- We introduce an Adaptive Token Expansion module that identifies and progressively expands tokens in challenging regions without directly accessing rendering errors.
- ATSplat achieves state-of-the-art rendering quality while using over 5.7× fewer Gaussians than previous dense pixel-aligned methods, demonstrating the importance of scene-adaptive capacity allocation for compact feed-forward 3DGS.

2 Related Work

2.1 3D Representations and Novel-View Synthesis

Neural radiance fields (NeRF) represent a scene as a continuous radiance-density field and synthesize novel views through volumetric rendering [Mildenhall et al. 2020]. A large body of work has improved NeRF in several aspects, which include anti-aliasing [Barron et al. 2021], training speed [Müller et al. 2022], explicit factorization [Chan et al. 2022; Chen et al. 2022], and generalizable radiance fields [Chen et al. 2021; Wang et al. 2021; Yu et al. 2021].

Despite their high fidelity, many neural-field methods require dense ray sampling and/or costly per-scene optimization. 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] instead represents scenes with a set of anisotropic Gaussian primitives and optimizes them via differentiable rasterization, enabling high-quality real-time rendering. A rapidly growing body of follow-up work has extended it along many directions, including surface reconstruction [Guédon and Lepetit 2024; Huang et al. 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]. A key strength of optimization-based 3DGS lies in adaptive primitive allocation based on scene complexity, which is enabled by 3D placement of Gaussians and adaptive density control.

2.2 Feed-forward 3D Gaussian Splatting

Instead of optimizing Gaussians per scene, recent feed-forward 3DGS methods directly predict Gaussian primitives from multi-view

captures, thereby addressing expensive per-scene optimization costs. 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] further 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. 2024], and unconstrained settings [Jiang et al. 2025]. Large reconstruction models (LRM) further scale this pixel-aligned paradigm with large transformer backbones [Tang et al. 2024; Xu et al. 2024; Zhang et al. 2024a]. More recently, iLRM [Kang et al. 2025] decouples the viewpoint tokens from input images and predicts Gaussians in a separate, low-resolution grid, yielding a more compact set of Gaussians. However, such uniform spatial downsampling reduces primitives across the entire scene regardless of scene complexity. This leads to under-allocated Gaussians in complicated regions and degrades rendering quality in these regions.

Although these methods achieve impressive reconstruction speed and rendering quality, most of them follow dense pixel-aligned formulations: the number of primitives are strongly tied to the image resolution and the number of views. As a result, simple and textureless regions can produce many redundant Gaussians, while challenging regions cannot receive additional capacity that they demand. ATSplat overcomes this limitation through sparse-to-adaptive capacity allocation, where our adaptive anchor tokens are selectively expanded based on reconstruction difficulty. Since this expansion proceeds inside the decoder, the expanded tokens are further refined by cross-attending to input images, allocating not only primitives but also computation to challenging regions.

2.3 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, 2025b]. 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. 2026]. 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 stages.

Concurrent to our work, an emerging line of works move more directly beyond pixel-aligned formulations. VolSplat [Wang et al. 2025a] replaces image-grid alignment with voxel-aligned prediction, and SparseSplat [Zhang et al. 2026a] adaptively samples Gaussians according to local information richness. Several works adaptively sample Gaussians from the image grid, with multi-granularity Gaussians [Kim et al. 2026] or adaptive locations within image grids [Moreau et al. 2025]. Other concurrent approaches decouple Gaussian prediction from pixels using 3D anchored feature volumes [Zhang et al. 2026b], or global learnable tokens [An et al. 2025; Itkin et al. 2026; Ren et al. 2026].

ATSplat is aligned with this emerging direction, but differs in how adaptive capacity is allocated. Instead of starting from a dense pixel-aligned Gaussian set, a fixed voxel lattice, or purely learnable global tokens, ATSplat constructs a scene-conditioned scaffold of sparse 3D anchor tokens from coarse depth and cameras. Each anchor is regressed into a set of local Gaussians with learnable 3D offsets, allowing primitives to be freely placed beyond camera rays. The Adaptive Token Expansion module identifies under-reconstructed tokens and selectively expands them. This sparse-to-adaptive formulation aims to fill the missing component in feed-forward 3DGS: allocating representation and computation capacity according to reconstruction difficulty, rather than input image structure.

3 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 multi-view images with camera poses $\{\mathbf{I}_v, \boldsymbol{\pi}_v\}_{v=1}^{V}$, a feed-forward 3DGS model predicts a set of Gaussian attributes:

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

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.

4 ATSplat Framework

We introduce ATSplat, a feed-forward 3DGS framework that restores the scene-adaptive Gaussian placement of per-scene 3DGS optimization. Our key idea is to recast three guiding principles of 3DGS–sparse initialization, free 3D placement, adaptive capacity allocation–as feed-forward operations centered on a set of 3D anchor tokens. Concretely, sparse initialization is achieved by unprojecting coarse patches into 3D, free 3D placement by regressing local primitive offsets from each anchor, and adaptive capacity allocation by expanding high-uncertainty anchors during decoding.

Fig. 2 (a) illustrates an overview of the framework. Given multi-view images, a multi-view encoder first extracts coarse patch features and predicts patch-level depths. The depths are unprojected

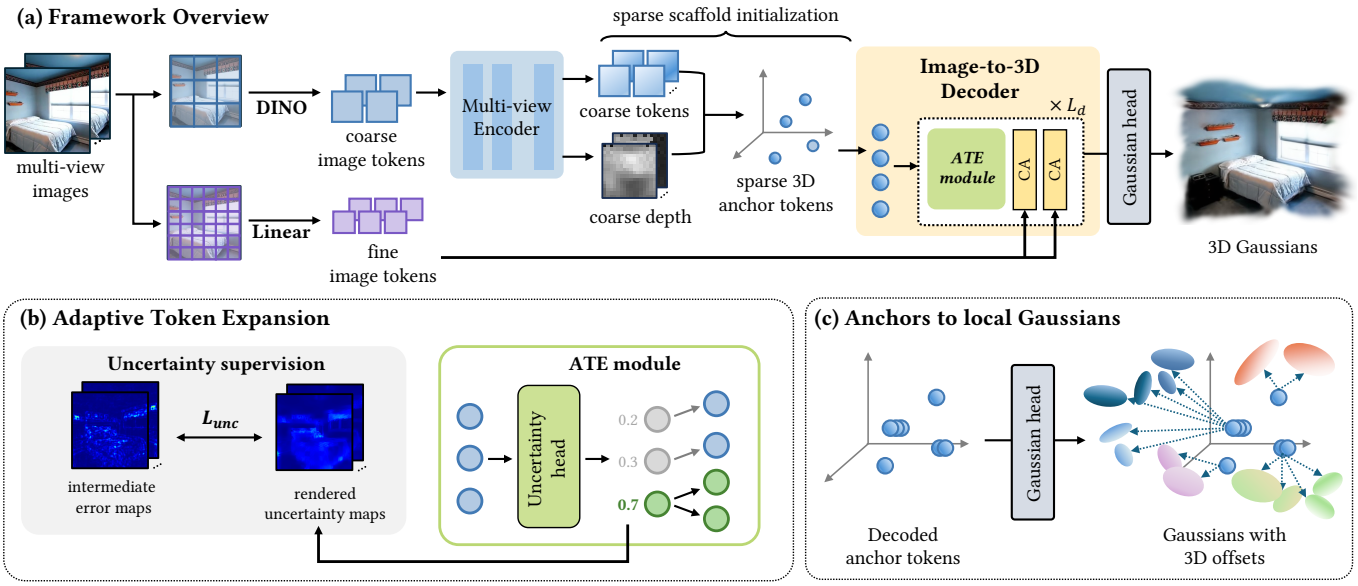

Fig. 2. **(a) Framework overview.** Given posed multi-view images, a multi-view image encoder extracts coarse patch features and estimates patch-level depths. The predicted depths and patch features are used to initialize a sparse set of 3D anchor tokens, serving as a sparse scene scaffold. An image-to-3D decoder refines these anchors through L_d decoder blocks, with an Adaptive Token Expansion (ATE) module that allocates additional capacity to under-reconstructed regions. A Gaussian head then decodes each refined anchor into a set of local 3D Gaussians. **(b) Adaptive Token Expansion.** At each decoder block, a lightweight MLP predicts a per-anchor uncertainty score, supervised by 2D error maps computed from intermediate Gaussians. Anchors with high uncertainty scores are expanded into multiple tokens, increasing representational capacity where needed. **(c) Anchors to local Gaussians.** Each refined anchor token is regressed into K local Gaussians whose centers are placed as 3D offsets relative to the anchor, decoupling primitive positions from the input pixel grid.

with the camera rays, assigning coarse patch features to sparse 3D locations that form an 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 associated with under-reconstructed regions, increasing representational 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. The following paragraphs describe details of each component.

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 frozen 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 anchor tokens by injecting fine-grained image information through cross-attention. 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 and cross-attention layers. The ATE module selectively expands anchors associated with under-reconstructed regions, progressively increasing representational capacity where needed. We describe details of ATE in Sec. 5.

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:

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

and each Gaussian center is placed relative to the anchor as

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

which places primitive positions freely from the input pixel grid.

5 Adaptive Token Expansion

Unlike per-scene optimization, a feed-forward model cannot directly access rendering errors or gradients during decoding, making

it non-trivial to identify under-reconstructed regions. Therefore, we approximate the rendering error associated with each anchor through a lightweight MLP, and use these predictions to selectively expand anchors that require additional representational capacity.

At each decoder block, we estimate an uncertainty 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 top fraction ρ_l of anchors with the highest scores as expansion targets. Each selected feature is passed through a linear projection that produces M residual features, yielding M child tokens that share the parent anchor coordinate $\mathbf{p}_i$:

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

The expanded tokens together with the unselected tokens are concatenated and passed to the next decoder block. This design identifies under-reconstructed regions in a single forward pass, without auxiliary renderings or back-propagation at inference.

Learning uncertainty in 2D space. To align the predicted uncertainty with the actual reconstruction difficulty, we supervise $g_\phi^{(l)}$ using 2D reconstruction errors. After the l-th decoder block, we apply the Gaussian head to the current anchors to obtain an intermediate Gaussian set $\mathcal{G}^{(l)}$. For each anchor i, the score $u_i^{(l)}$ is attached as a scalar attribute to its K Gaussians and rasterized via standard alpha-compositing, producing a 2D uncertainty map $\hat{\mathbf{U}}^{(l)}$. We supervise $\hat{\mathbf{U}}^{(l)}$ against the error map $\mathbf{U}^{(l)}$ of $\mathcal{G}^{(l)}$:

$$\mathcal{L}_{\text{unc}}^{(l)} \;=\; \left\| \hat{\mathbf{U}}^{(l)} - \text{sg}\big(\mathbf{U}^{(l)}\big) \right\|_1, \tag{8}$$

where $\text{sg}(\cdot)$ stops gradients into the Gaussian parameters so that this loss updates only $g_\phi^{(l)}$. The error map $\mathbf{U}^{(l)}$ is computed using the rendered results $\mathcal{G}^{(l)}$ and ground-truth images. We empirically use D-SSIM to compute these intermediate error maps.

6 Training

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

$$\mathcal{L}_{\text{render}} \;=\; \mathcal{L}_{\text{MSE}} + \lambda_{\text{p}} \cdot \mathcal{L}_{\text{perceptual}}, \tag{9}$$

where we use $\lambda_p = 0.5$. To train the uncertainty 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

$$\mathcal{L} \;=\; \mathcal{L}_{\text{render}} \;+\; \sum_{l=1}^{L} \big(\lambda_{\text{int}} \cdot \mathcal{L}_{\text{interm}}^{(l)} \;+\; \lambda_{\text{unc}} \cdot \mathcal{L}_{\text{unc}}^{(l)},\big). \tag{10}$$

where we use $\lambda_{\text{interm}} = 0.5$ and $\lambda_{\text{unc}} = 0.1$ in all experiments.

7 Experiments

7.1 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 larger-scale captures of both 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 steps and train/test splits.

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 codes for this setting.

Method	PSNR↑	SSIM↑	LPIPS↓	#Gauss.↓	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
GS-LRM [Zhang et al. 2024a]	28.10	0.892	0.114	131K	-
DepthSplat [Xu et al. 2025]	27.47	0.889	0.114	131K	0.067
LongLRM [Ziwen et al. 2025]	<u>28.54</u>	0.895	**0.109**	131K	-
iLRM [Kang et al. 2025]	**28.65**	<u>0.900</u>	<u>0.110</u>	131K	0.025
TokenGS [Ren et al. 2026]	28.41	**0.903**	0.135	262K	0.066
Ours	28.46	0.894	0.118	**23K**	**0.022**

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. Runtime is measured on a single RTX 3090 GPU unless otherwise specified. For RealEstate10K, we use the evaluation viewpoint indices provided in pixelSplat [Charatan et al. 2024]. For DL3DV, we follow the protocol of DepthSplat [Xu et al. 2025] and evaluate under 2, 4, 6 input-view setups. We additionally evaluate results under varying viewpoint overlaps following NoPoSplat [Ye et al. 2025b] in the supplementary.

Baselines. We compare ATSplat with recent feed-forward 3DGS methods, including pixelSplat [Charatan et al. 2024], MVSplat [Chen et al. 2024a], DepthSplat [Xu et al. 2025], NoPoSplat [Ye et al. 2025b], and iLRM [Kang et al. 2025], where applicable. For quantitative comparisons, we use reported numbers when methods follow the same evaluation protocol, and otherwise evaluate available public checkpoints under the same evaluation setup. For qualitative comparisons on DL3DV, we compare primarily with DepthSplat, as it is the only feed-forward 3DGS baseline with a publicly available checkpoint under this setting. For high-resolution evaluations, we further compare with 3DGS [Kerbl et al. 2023] and Mip-Splatting [Yu et al. 2024a] as optimization-based baselines. These methods are initialized with sparse SfM [Schonberger and Frahm 2016] points of the input views and optimized for 30K iterations.

Implementation details. ATSplat consists of a multi-view encoder with 12 global self-attention layers, followed by a decoder with 4 blocks. We use a frozen DINOv2-B [Oquab et al. 2023] model with a 768 dimension for the coarse patch branch. An ATE module is employed in all decoder blocks except the first one, with token selection ratios ρ_l of 0.5, 0.5, and 0.25, respectively, and an expansion ratio of 2. Each anchor token is finally decoded into $K = 16$ local Gaussians. ATSplat is first trained on RealEstate10K with 2 input views using 4 RTX 4090 GPUs for about two days, serving as the base model. It is further trained on DL3DV with 6 input views using 4 H200 GPUs for two days. For the high-resolution setting, we further fine-tune this model with 10 input views using 8 H200 GPUs for less than two days. Please refer to the supplementary for more details.

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 predicts pixel-aligned Gaussians at 2× downsampled resolution, which results in 4× fewer Gaussians.

	2 input views				4 input views				6 input views			
Method	PSNR↑	SSIM↑	LPIPS↓	#Gauss.↓	PSNR↑	SSIM↑	LPIPS↓	#Gauss.↓	PSNR↑	SSIM↑	LPIPS↓	#Gauss.↓
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. 2025]	–	–	–	–	–	–	–	–	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. 2026a]	–	–	–	–	–	–	–	–	24.20	0.817	0.168	150K
Ours	**20.07**	**0.630**	0.332	**40K**	**25.67**	**0.827**	**0.172**	**80K**	**27.28**	**0.868**	**0.138**	**120K**

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. *: Runtime of DepthSplat is measured using a single A6000 GPU due to out-of-memory.

Method	PSNR↑	SSIM↑	LPIPS↓	#Gauss.↓	Time(s)↓
3DGS [Kerbl et al. 2023]	22.87	0.758	**0.229**	553K	> 10 m
Mip-Splatting [Yu et al. 2024a]	22.52	0.746	0.240	676K	> 10 m
DepthSplat* [Xu et al. 2025]	21.33	0.740	0.270	5898K	0.758 s
iLRM [Kang et al. 2025]	24.35	0.781	0.256	1474K	0.700 s
Ours	**24.85**	**0.794**	0.241	**311K**	**0.677 s**

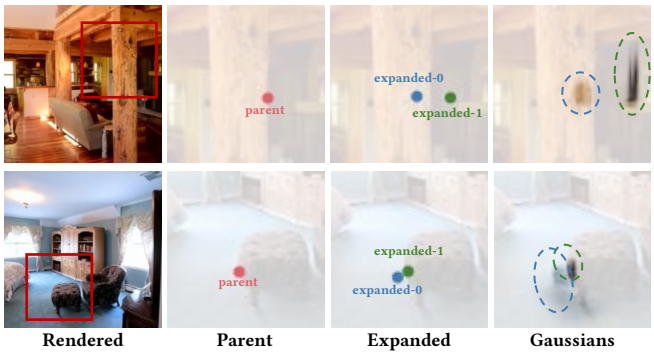

Fig. 3. **Visualization of expanded tokens.** The coordinate of each expanded token is computed as the mean center of its decoded Gaussians.

7.2 Results

We first evaluate ATSplat on common benchmark setups adopted in many previous works [Charatan et al. 2024; Chen et al. 2024a; Kang et al. 2025; 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. We further encourage the readers to check the supplementary for more qualitative results and video comparisons.

RealEstate10K. Tab. 1 reports two-view results on RealEstate10K. ATSplat achieves rendering quality comparable to state-of-the-art feed-forward methods while using only 23K Gaussians, corresponding to a 5.7× reduction compared to dense pixel-aligned formulations. Fig. 5 further confirms this, showing accurate reconstruction of fine details and geometry. By allocating primitives from sparse anchors and selectively expanding them, ATSplat maintains high-quality results with a much smaller representation compared to dense pixel-aligned baselines. We further provide evaluations with varying viewpoint overlaps in the supplementary, which highlights the efficacy of our 3D anchor designs.

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 outperforms prior feed-forward baselines while using substantially fewer Gaussians, demonstrating that the sparse-to-adaptive formulation remains effective in larger and more diverse scenes. Fig. 6 also supports this observation, with thin structures and intricate geometry faithfully recovered. In particular, iLRM [Kang et al. 2025] reduces the number of Gaussians by predicting them at a lower spatial resolution. However, this uniform reduction decreases Gaussian density across the entire scene, under-allocating capacity to regions with fine structures or intricate geometry. In contrast, ATSplat begins with sparse anchors and selectively expands them in uncertain regions, achieving high-quality results with a compact Gaussian set.

7.3 High-Resolution Novel-View Synthesis

We further evaluate ATSplat on DL3DV at 512 × 960 resolution to demonstrate that our sparse-to-adaptive design scales to high-resolution rendering. For efficiency, we downsample the images used by the coarse patch branch (*i.e.*, DINO) by 2×. We also increase the selection ratio ρ_1 of the first ATE block from 0.5 to 0.8, as the finer details exposed at this resolution benefit from more expansion. We compare with available feed-forward 3DGS methods as well as optimization-based 3DGS baselines. For DepthSplat [Xu et al. 2025], we use the public checkpoint trained at 448 × 768 resolution, the only configuration with released weights applicable to this setup.

As shown in Tab. 3, ATSplat reconstructs high-resolution scenes in a single forward pass and completes inference in less than one second. It achieves compelling rendering results while using far fewer Gaussians than previous dense pixel-aligned feed-forward methods, which can also be found in Fig. 7. This efficiency becomes important at high resolution, where pixel-aligned methods typically produces more than a million Gaussians.

7.4 Analysis

To better understand the behaviors of our core components, we provide analyses on expanded tokens and uncertainty predictions.

Table 4. **Ablation results on the 3D anchor designs.**

Anchor init.	Gaussian center	PSNR↑	SSIM↑	LPIPS↓
None	Ray + depth	27.35	0.874	0.138
Learnable	Direct xyz	24.62	0.802	0.204
None	Direct xyz	20.87	0.650	0.364
Patch depth	Anchor + offset	**28.46**	**0.894**	**0.118**

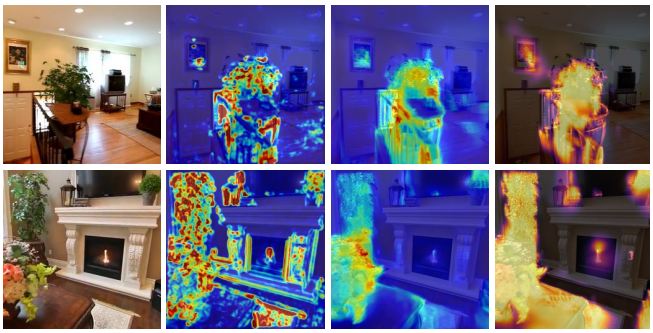

Fig. 4. **Visualization of uncertainty scores and selected tokens** at the last block. We visualize actual error maps (D-SSIM), predicted uncertainty maps, and alpha maps of Gaussians decoded from the selected tokens.

Behavior of expanded tokens. We first analyze how ATE increases local representation capacity. For each expanded token, we compute a representative 3D location by averaging the centers of the Gaussians decoded from that token. Although ATE does not explicitly update anchor coordinates during expansion, the expanded token features produce distinct local Gaussian offsets and features. As shown in Fig. 3, child tokens spread around their parent tokens and cover different nearby regions instead of collapsing to the same position. This supports that ATE learns to refine local geometry and appearance based on the detailed scene content.

Predicted uncertainty and selected tokens. We further analyze whether the learned uncertainty provides a meaningful proxy for reconstruction difficulty. As shown in Fig. 4, the predicted uncertainty is high around regions with larger rendering errors, including fine structures and visually complex areas. The selected tokens are concentrated in these uncertain regions, indicating that ATE successfully identifies where additional Gaussians are needed and places more capacity to these regions with diversified expanded tokens.

7.5 Ablations

We examine the effects of our two core components through the ablations: 3D anchor designs and uncertainty-guided selection. All ablations are conducted on RealEstate10K with two input views.

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. 4, 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 scene structure, while local offsets preserve the flexibility needed for accurate Gaussian placement.

Table 5. **Ablation results on the ATE module.**

Method	PSNR↑	SSIM↑	LPIPS↓
Random	27.97	0.888	0.126
Farthest Point Sampling (FPS)	27.98	0.888	0.125
Fully learnable (STE) [Yin et al. 2019]	27.93	0.887	0.125
No expansion	27.02	0.868	0.153
Uncertainty	**28.46**	**0.894**	**0.118**

ATE module. We further ablate the selection strategy and expansion in ATE. All selection-based variants use the same expansion ratio and produce the same number of Gaussians. As shown in Tab. 5, random selection and FPS provide limited gains as they allocate capacity without targeting difficult regions. We also compare with a straight-through estimator (STE) [Yin et al. 2019], following a prior upsampling design [Nam et al. 2025], but find that it does not improve over simple heuristic selection. Removing token expansion entirely leads to a notable performance drop. Our full design achieves the best performance, demonstrating the effectiveness of both uncertainty-guided selection and learnable expansion.

8 Conclusion and Future Works

We presented ATSplat, a feed-forward 3D Gaussian Splatting framework that restores the scene-adaptive capacity allocation of the 3DGS pipeline. Instead of densely assigning Gaussians to input image grids, ATSplat builds a sparse set of adaptive 3D anchor tokens, decodes them into a set of local Gaussians with 3D relative offsets, and progressively selects and expands tokens associated with challenging regions. This sparse-to-adaptive formulation decouples Gaussian placement from input pixels and allows us to allocate representational capacity according to scene complexity. ATSplat reinterprets the core design principles of 3DGS within a feed-forward framework, achieving state-of-the-art quality with more than 5.7× fewer Gaussians.

While ATSplat addresses the scene-adaptive capacity allocation problem in feed-forward 3DGS, several directions remain open for future work. First, ATSplat expands tokens adaptively but does not prune those that become redundant during decoding. A pruning mechanism analogous to that of 3DGS could be a key for improving decoder efficiency and more effective capacity allocation. Second, several architectural improvements can be made for more scalable designs, to extend ATSplat to larger-scale scenes, more input views, and higher resolutions. Finally, extending ATSplat to unposed settings would broaden its applicability to in-the-wild images.

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. 2025. C3G: Learning Compact 3D Representations with 2K Gaussians. *arXiv preprint arXiv:2512.04021* (2025).

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. 2022. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 16123–16133.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. 2024. 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)*. 19457–19467.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. TensoRF: Tensorial Radiance Fields. In *European Conference on Computer Vision (ECCV)*.

Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. 2021. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF international conference on computer vision*. 14124–14133.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. 2024a. MVSplat: Efficient 3D Gaussian Splatting from Sparse Multi-View Images. In *European Conference on Computer Vision (ECCV)*.

Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. 2024b. Mvsplat360: Feed-forward 360 scene synthesis from sparse views. *Advances in Neural Information Processing Systems* 37 (2024), 107064–107086.

Antoine Guédon and Vincent Lepetit. 2024. 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*. 5354–5363.

Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jisang Han, Jiaolong Yang, Chong Luo, and Seungryong Kim. 2024. Pf3plat: Pose-free feed-forward 3d gaussian splatting. *arXiv preprint arXiv:2410.22128* (2024).

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*. 1–11.

Roni Itkin, Noam Issachar, Yehonatan Keypur, Xingyu Chen, Anpei Chen, and Sagie Benaim. 2026. GlobalSplat: Efficient Feed-Forward 3D Gaussian Splatting via Global Scene Tokens. *arXiv preprint arXiv:2604.15284* (2026).

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

Gyeongjin Kang, Seungtae Nam, Xiangyu Sun, Sameh Khamis, Abdelrahman Mohamed, and Eunbyung Park. 2025. iLRM: An Iterative Large 3D Reconstruction Model. *arXiv preprint arXiv:2507.23277* (2025).

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

Injae Kim, Chaehyeon Kim, Minseong Bae, Minseok Joo, and Hyunwoo J. Kim. 2026. F4Splat: Feed-Forward Predictive Densification for Feed-Forward 3D Gaussian Splatting. *arXiv preprint arXiv:2603.21304* (2026).

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. 2024. 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*. 22160–22169.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 20654–20664.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision (ECCV)*.

Arthur Moreau, Richard Shaw, Michal Nazarczuk, Jisu Shin, Thomas Tanay, Zhensong Zhang, Songcen Xu, and Eduardo Pérez-Pellitero. 2025. Off The Grid: Detection of Primitives for Feed-Forward 3D Gaussian Splatting. *arXiv preprint arXiv:2512.15508* (2025).

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics (SIGGRAPH)* 41, 4 (2022), 102:1–102:15. doi:10.1145/3528223.3530127

Seungtae Nam, Xiangyu Sun, Gyeongjin Kang, Younggeun Lee, Seungjun Oh, and Eunbyung Park. 2025. Generative densification: Learning to densify gaussians for high-fidelity generalizable 3d reconstruction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 26683–26693.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193* (2023).

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

Jiawei Ren, Michal Jan Tyszkiewicz, Jiahui Huang, and Zan Gojcic. 2026. TokenGS: Decoupling 3D Gaussian Prediction from Pixels with Learnable Tokens. *arXiv preprint arXiv:2604.15239* (2026).

Johannes L Schonberger and Jan-Michael Frahm. 2016. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4104–4113.

Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. 2024. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10208–10217.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. 2024. LGM: Large Multi-View Gaussian Model for High-Resolution 3D Content Creation. In *European Conference on Computer Vision (ECCV)*.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. 2021. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4690–4699.

Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Feng Chen, Zheng Zhu, Donny Y. Chen, and Bohan Zhuang. 2025a. VolSplat: Rethinking Feed-Forward 3D Gaussian Splatting with Voxel-Aligned Prediction. *arXiv preprint arXiv:2509.19297* (2025).

Yiming Wang, Lucy Chai, Xuan Luo, Michael Niemeyer, Manuel Lagunas, Stephen Lombardi, Siyu Tang, and Tiancheng Sun. 2026. Learning Efficient Fuse-and-Refine for Feed-Forward 3D Gaussian Splatting. *Advances in Neural Information Processing Systems* 38 (2026), 76311–76341.

Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. 2024. Freesplat: Generalizable 3d gaussian splatting towards free view synthesis of indoor scenes. *Advances in Neural Information Processing Systems* 37 (2024), 107326–107349.

Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. 2025b. Freesplat++: Generalizable 3d gaussian splatting for efficient indoor scene reconstruction. *arXiv preprint arXiv:2503.22986* (2025).

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 20310–20320.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. 2025. DepthSplat: Connecting Gaussian Splatting and Depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16453–16463.

Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. 2024. GRM: Large Gaussian Reconstruction Model for Efficient 3D Reconstruction and Generation. In *European Conference on Computer Vision (ECCV)*.

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. 2024. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *International Conference on Learning Representations*, Vol. 2024. 9142–9159.

Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. 2025a. YoNoSplat: You Only Need One Model for Feedforward 3D Gaussian Splatting. *arXiv preprint arXiv:2511.07321* (2025).

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. 2025b. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. In *International Conference on Learning Representations*, Vol. 2025. 54009–54033.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. 2019. Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets. In *International Conference on Learning Representations*.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2021. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4578–4587.

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

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

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. 2024a. GS-LRM: Large Reconstruction Model for 3D Gaussian Splatting. In *European Conference on Computer Vision (ECCV)*.

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

Shengjun Zhang, Xin Fei, Fangfu Liu, Haixu Song, and Yueqi Duan. 2024b. Gaussian graph network: Learning efficient and generalizable gaussian representations from multi-view images. *Advances in Neural Information Processing Systems* 37 (2024), 50361–50380.

Xiaoxue Zhang, Xiaoxu Zheng, Yixuan Yin, Tiao Zhao, Kaihua Tang, Michael Bi Mi, Zhan Xu, and Dave Zhenyu Chen. 2026b. AnchorSplat: Feed-Forward 3D Gaussian SplattingWith 3D Geometric Priors. *arXiv preprint arXiv:2604.07053* (2026).

Zicheng Zhang, Xiangting Meng, Ke Wu, and Wenchao Ding. 2026a. 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)*.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. 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. 2025. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4349–4359.

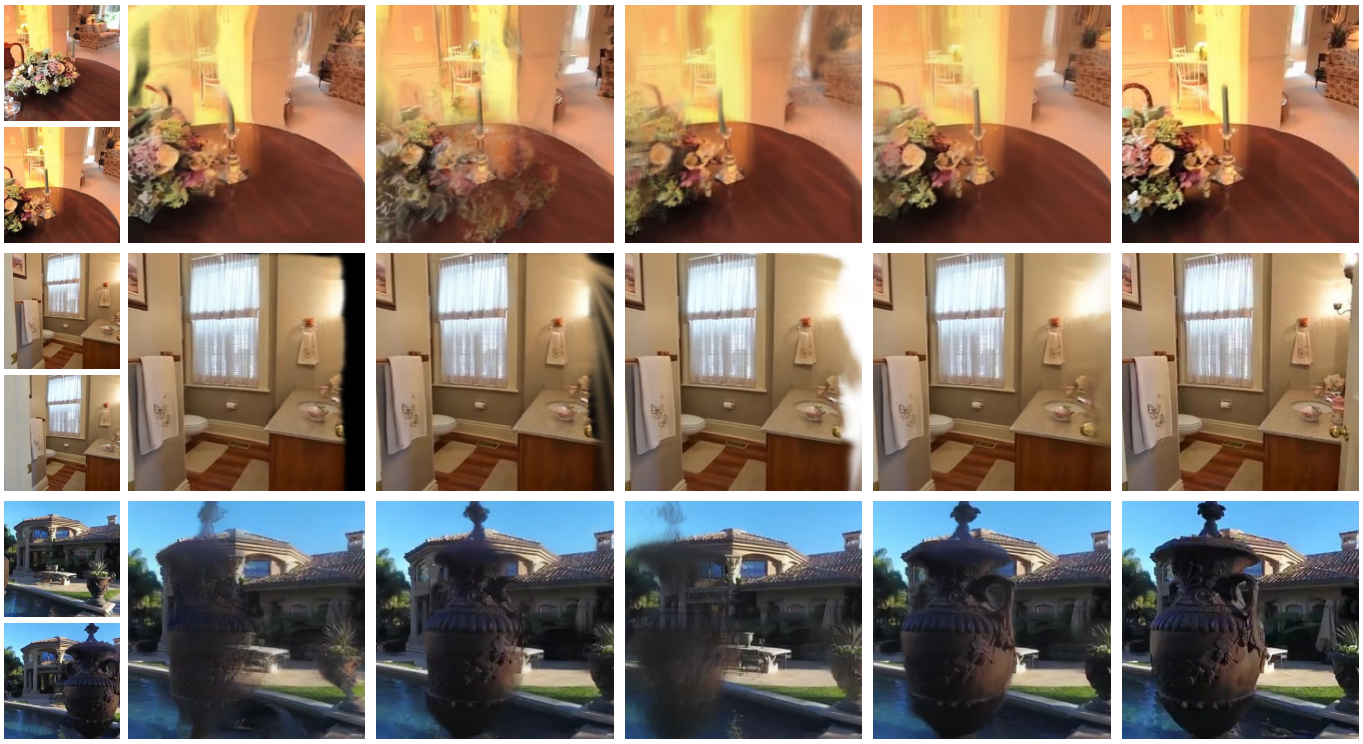

Fig. 5. **Qualitative comparisons on RealEstate10K at** 256 × 256 **resolution** with 2 input views. ATSplat reconstructs fine details in challenging regions while accurately reconstructing geometry, while using 5.7× fewer Gaussians compared to the dense pixel-aligned baselines.

Fig. 6. **Qualitative comparisons on DL3DV at** 256 × 448 **resolution** with 6 input views. ATSplat produces high-fidelity renderings with only 120K Gaussians, which is 5.7× fewer than DepthSplat. It also faithfully reconstructs thin structures, intricate details, and view-dependent appearance.

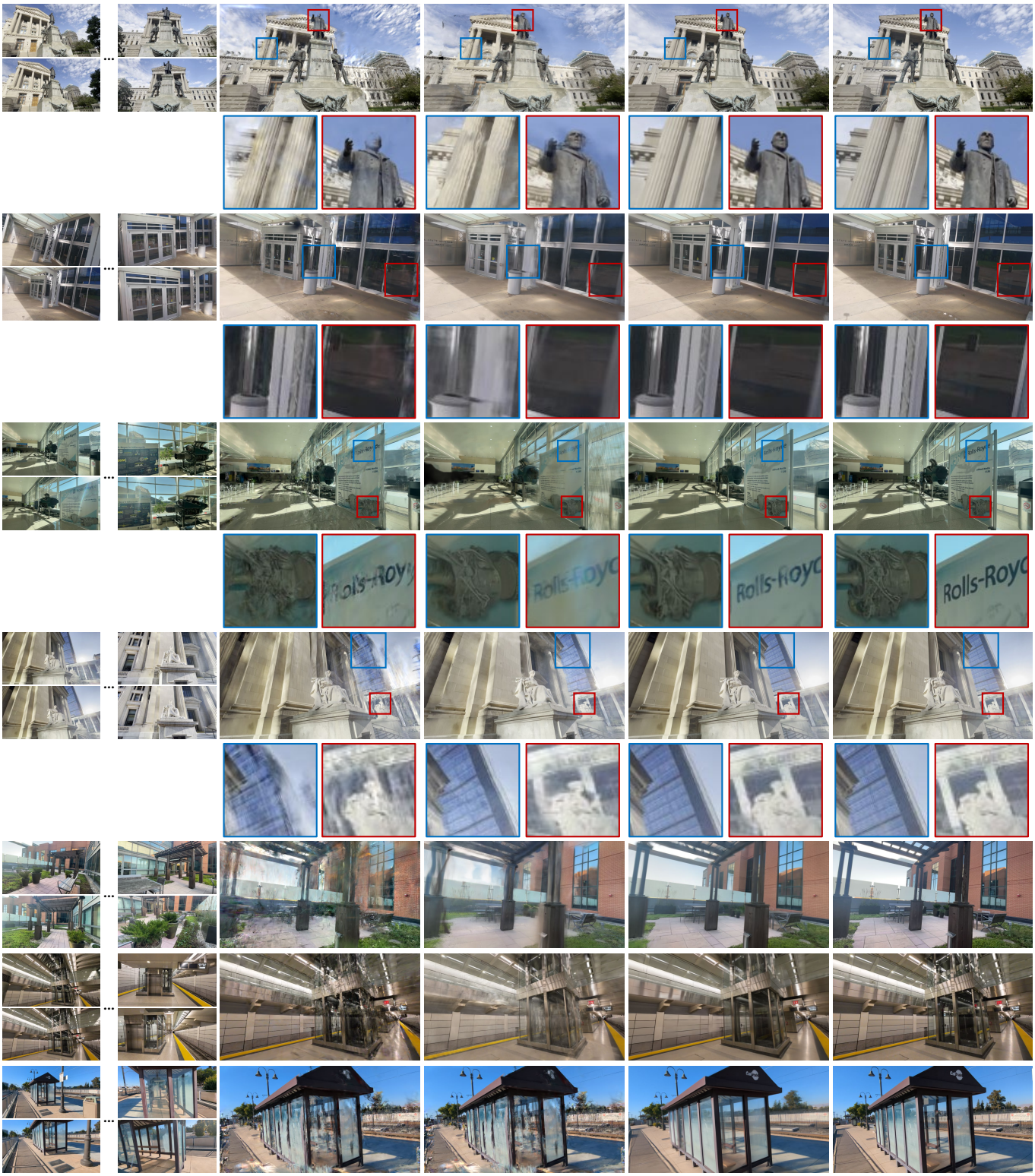

Fig. 7. **Qualitative comparisons on DL3DV at** 512 × 960 **resolution** with 12 input views. ATSplat recovers fine details and thin structures in less than a second, with a significantly reduced number of Gaussians, whereas 3DGS requires over 10 minutes of per-scene optimization to obtain these results.

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. ATSplat is trained in BF16 precision, which reduces memory consumption while maintaining stable optimization in our experiments. We optimize the model using AdamW [Loshchilov and Hutter 2017] with a weight decay of 0.05. The initial learning rate is set to 2×10^{-4} and is scheduled using cosine annealing after an initial warm-up phase. We use a batch size of 8 per GPU except for 256×448 DL3DV [Ling et al. 2024] experiments, which results in a total batch size of 32 for RealEstate10K [Zhou et al. 2018] and 64 for DL3DV experiments.

A.2 Architecture Details

We use a feature dimension of 768 throughout the model. The fine branch uses a patch size of 8, while the coarse branch uses a patch size of 14, following the resolution of DINO backbone. To ensure that the coarse and fine branches operate at a consistent relative resolution, we first downsample the input images before passing them to the DINO encoder in the coarse branch. This makes the spatial resolution of the fine patch grid twice that of the coarse patch grid. Each decoder block leverages standard cross-attention layers. Each of these layers consists of a cross-attention, a self-attention, and a feed-forward module. We employ one such layer in the first two blocks, and two layers in the others.

In addition to DINO features, we additionally patchify the raw images using a patch size of 16 and fuse the resulting image patch features to the corresponding DINO features through concatenation and a linear projection layer, after projecting them to the model dimension. For the fine branch, before feeding to the decoder, we also inject the encoder output features as coarse-level context, into the fine-resolution patch features. The final coarse patch features are upsampled to the spatial resolution of the fine patch grid and added to the fine patch features.

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. For the uncertainty head, we apply a softplus activation to the output. We empirically found this parameterization to be more stable in our setup. We also normalize the input camera poses before feeding them to the network. Given the input camera sequence, we compute the average camera pose and transform the camera coordinate system such that the average pose is aligned with the identity pose, following previous works [Kang et al. 2025].

A.3 Total Gaussian Budgets

All results reported in the main paper are obtained with the fixed selection ratios ρ_l. Under this setting, the number of tokens after the decoder is mainly determined by the input configuration. Given N_{init} initial anchor tokens, the number of final tokens is

$$N_{\text{final}} \;=\; N_{\text{init}} \prod_{l=1}^{L_{\text{ATE}}} \big(1 + \rho_l (M-1)\big), \tag{11}$$

where L_{ATE} denotes the number of decoder blocks equipped with an ATE module and M is the expansion ratio. Each token is then decoded into K local Gaussians, which yields $G = KN_{\text{final}}$ primitives in total. Tab. 6 lists the resulting token and Gaussian counts for every configuration used in our experiments. We emphasize that fixed ratios are not the only choice at inference, and we further report per-scene dynamic budgets obtained by uncertainty thresholding in Sec. C.1.

Table 6. **Token and Gaussian counts** under our experimental configurations. Each anchor token is decoded into $K = 16$ Gaussians, and the total number of Gaussians is $G = KN_{\text{final}}$. *: the first ATE block uses a selection ratio of 0.8 instead of 0.5 in this setting.

Dataset	Resolution	Views	N_{init}	N_{final}	G
RE10K	256×256	2	512	1440	23040
DL3DV	256×448	2	896	2520	40320
DL3DV	256×448	4	1792	5040	80640
DL3DV	256×448	6	2688	7560	120960
DL3DV	$512 \times 960^*$	12	5760	19440	311040

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. 2025b]. Small overlap typically corresponds to wider camera baselines, whereas large overlap corresponds to narrow baselines where most target viewpoints are placed close to the input views. Tab. 7 reports PSNR, SSIM, and LPIPS on RealEstate10K across the resulting overlap bins.

ATSplat attains the strongest performance in the small-overlap regime by a clear margin, becomes comparable to pixel-aligned baselines under medium overlap, and eventually falls behind in the large-overlap setting. This trend is a direct consequence of the differing formulations of ATSplat and pixel-aligned methods. Pixel-aligned methods predict per-pixel Gaussians anchored on input camera rays, and are therefore well-suited to large-overlap settings where most of the target content is directly observable from the input views. As overlap decreases, however, an increasing portion of the target view lies away from the input rays, and ray-constrained placement becomes a fundamental limitation. Our 3D anchor-offset effectively addresses this issue and achieves compelling results in the more challenging setup, thanks to our anchor-offset design that can place primitives freely in 3D space beyond input camera rays.

B.2 Results with Square-Cropped Inputs

We further compare ATSplat with YoNoSplat [Ye et al. 2025a], a concurrent feed-forward method evaluated on square-cropped images. To match its configuration, we train and evaluate ATSplat on DL3DV at 224×224 resolution with 6 input views, where both input and target images are square-cropped. As shown in Tab. 8, ATSplat

Table 7. **Quantitative comparisons on RE10K** with varying viewpoint overlaps. Smaller viewpoint overlaps indicate larger camera baseline ranges.

		Small			Medium			Large			Average		
Method	#Gauss.↓	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. 2025]	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**
Ours	**23K**	**24.09**	**0.815**	0.185	**26.56**	0.861	0.144	29.13	0.900	0.114	**26.70**	0.861	0.146

Table 8. **Quantitative comparisons on DL3DV at** 224×224 **resolution** with 6 input views. *: the number of Gaussians is averaged over the test scenes due to the opacity-based Gaussian pruning.

Method	PSNR↑	SSIM↑	LPIPS↓	#Gauss.↓
YoNoSplat [Ye et al. 2025a]	24.72	0.817	0.139	212K*
Ours	**27.97**	**0.882**	**0.118**	**52K**

Table 9. **Zero-shot transfer from RealEstate10K to DL3DV** at 256×256 resolution with 2 input views. All methods are trained on RealEstate10K and evaluated on DL3DV without fine-tuning.

Method	PSNR↑	SSIM↑	LPIPS↓	#Gauss.↓
MVSplat [Chen et al. 2024a]	16.18	0.459	0.441	131K
DepthSplat [Xu et al. 2025]	18.07	0.571	**0.352**	131K
iLRM [Kang et al. 2025]	18.82	**0.588**	**0.352**	131K
Ours	**19.17**	0.584	0.371	**23K**

Table 10. **Extrapolated novel-view synthesis on DL3DV** with 6 input views. Target views are sampled from the 10 frames placed before and after the input view window.

Method	PSNR↑	SSIM↑	LPIPS↓
DepthSplat [Xu et al. 2025]	19.22	0.687	0.265
Ours	**22.11**	**0.747**	**0.246**

achieves higher quality across all metrics while using 4.1× fewer Gaussians on average.

B.3 Cross-Dataset Generalization

We next examine how our formulation transfers to unseen data. We evaluate all models trained on RealEstate10K directly on DL3DV at 256×256 resolution with 2 input views, without any fine-tuning. Tab. 9 reports the zero-shot transfer results. ATSplat attains the best PSNR and comparable SSIM and LPIPS while using significantly fewer Gaussians, indicating that ATSplat well adapts to unseen data distributions.

B.4 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 placed before and after the input view window, so that a large portion of the target content is not directly observable from the input rays. Tab. 10 compares ATSplat with DepthSplat [Xu et al. 2025]. This setting shares the same difficulty as the small-overlap regime in Sec. B.1, where ray-constrained placement cannot cover content away from the input rays. ATSplat adaptively places primitives freely in 3D space, and thus achieves better results as the target viewpoints move away from the input views.

Table 11. **Comparison of separately trained models and a single unified model** jointly trained across both datasets and all input view counts. RE10K is evaluated at 256×256 and DL3DV at 256×448 resolution.

	Separate			Unified		
Setting	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
RE10K, 2 views	28.46	0.894	0.118	28.11	0.890	0.125
DL3DV, 2 views	20.07	0.630	0.332	21.80	0.702	0.283
DL3DV, 4 views	25.67	0.827	0.172	25.73	0.830	0.169
DL3DV, 6 views	27.28	0.868	0.138	27.26	0.868	0.136

Table 12. **Per-scene dynamic budgets on RealEstate10K** with 2 input views. Each threshold triplet lists the uncertainty thresholds applied at the three ATE blocks, replacing the fixed selection ratios at inference.

				#Gauss.↓		
Selection	PSNR↑	SSIM↑	LPIPS↓	Avg.	Min.	Max.
Fixed ratios	28.46	0.894	0.118	23K	23K	23K
0.1 / 0.1 / 0.4	27.90	0.888	0.127	13K	8K	36K
0.05 / 0.05 / 0.2	28.35	0.895	0.118	19K	8K	55K
0.01 / 0.01 / 0.04	28.64	0.899	0.112	39K	14K	65K

B.5 A Unified Model across Datasets and View Counts

Following previous feed-forward methods [Charatan et al. 2024; Chen et al. 2024a; Xu et al. 2025], we train a separate model for each resolution. This choice follows the common evaluation protocol for fair comparisons and is not a property of ATSplat, as none of its components is tied to a specific resolution or number of input views. To verify this, we train a single model jointly on both datasets across all input view counts, and evaluate it under each setup. Tab. 11 compares the unified model with the separately trained ones. The unified model achieves comparable quality with the separately trained models, verifying that ATSplat can be trained jointly across datasets and view counts.

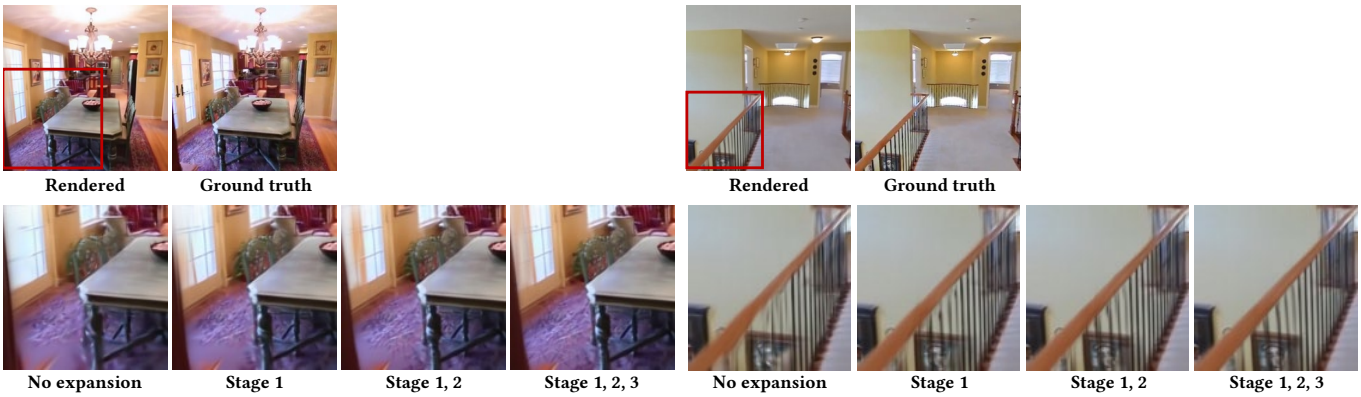

Fig. 8. **Effect of progressively enabling token expansion across decoder blocks.** Starting from a configuration with expansion disabled at all decoder blocks, we gradually enable expansion from the earliest block to the latest. As more blocks are activated, fine-grained texture and thin structures are progressively recovered.

Table 13. **Effect of the number of Gaussians per anchor token** K on RealEstate10K with 2 input views. *: our default setting used in all other experiments.

K	PSNR↑	SSIM↑	LPIPS↓	#Gauss.↓
4	27.63	0.877	0.141	5K
8	28.25	0.890	0.124	11K
16*	28.46	0.894	0.118	23K
32	28.66	0.898	0.113	46K
64	28.73	0.899	0.111	92K

Table 14. **Ablation results on the 3D anchor designs** on DL3DV with 6 input views.

Anchor init.	Gaussian center	PSNR↑	SSIM↑	LPIPS↓
None	Ray + depth	26.37	0.845	0.161
Learnable	Direct xyz	21.05	0.625	0.359
None	Direct xyz		Diverged	
Patch depth	Anchor + offset	**27.28**	**0.868**	**0.138**

Table 15. **Ablation results on the ATE module** on DL3DV with 6 input views.

Method	PSNR↑	SSIM↑	LPIPS↓
Random	26.31	0.846	0.154
Farthest point sampling (FPS)	26.21	0.843	0.158
Fully learnable (STE) [Yin et al. 2019]	26.72	0.859	0.143
No expansion	25.17	0.810	0.200
Uncertainty	**27.28**	**0.868**	**0.138**

C Additional Analysis and Ablations

C.1 Dynamic Gaussian Budgets via Uncertainty Thresholds

With fixed selection ratios, ATSplat distributes primitives adaptively within a budget that is determined by the input configuration, rather than selecting the budget itself. The same model can instead expand every token whose uncertainty exceeds certain thresholds, which produces scene-dependent budgets at inference without retraining or architectural changes.

As shown in Tab. 12, the resulting budget varies by up to 6.9× across scenes, indicating that it now follows scene complexity rather than the input configuration. Lower thresholds expand more tokens and improve quality, and only the smallest reaches the state-of-the-art quality, still using 3.4× fewer Gaussians compared to iLRM.

C.2 Scaling the Gaussian Budget

The number of Gaussians decoded from each anchor token, K, can control the total Gaussian budget of ATSplat. We vary K from 4 to 64 on RealEstate10K with 2 input views, keeping all other settings identical. As shown in Tab. 13, rendering quality improves consistently as K grows, with diminishing returns beyond our default setting of $K = 16$.

C.3 Expansion at each decoder block

To gain deeper insights into the ATE module, we further analyze its behavior at each decoder block by progressively enabling expansion from the earliest block to the latest, while keeping all other settings identical. Fig. 8 shows the resulting renderings. With expansion fully disabled, the limited representation capacity leaves some fine details missing. As expansion is enabled at additional blocks, the renderings are progressively refined, with fine-grained texture and high-frequency detail recovered more faithfully.

C.4 Ablations on DL3DV

The ablations in the main paper are conducted on RealEstate10K with 2 input views. We further validate our core components on the DL3DV dataset with 6 input views, to examine whether they remain effective on larger and more complex scenes. Tab. 14 and Tab. 15 report the results for the 3D anchor designs and the ATE

module, respectively. On the DL3DV dataset, regressing absolute Gaussian centers without any anchor initialization fails to converge, indicating that coarse 3D anchors become more critical as the spatial extent of the scene grows.

D Additional Qualitative Results

We provide additional qualitative comparisons on RealEstate10K and DL3DV. Fig. 9 presents results on RealEstate10K. Fig. 10 and Fig. 11 present results on the DL3DV dataset at 256×448 and 512×960 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.

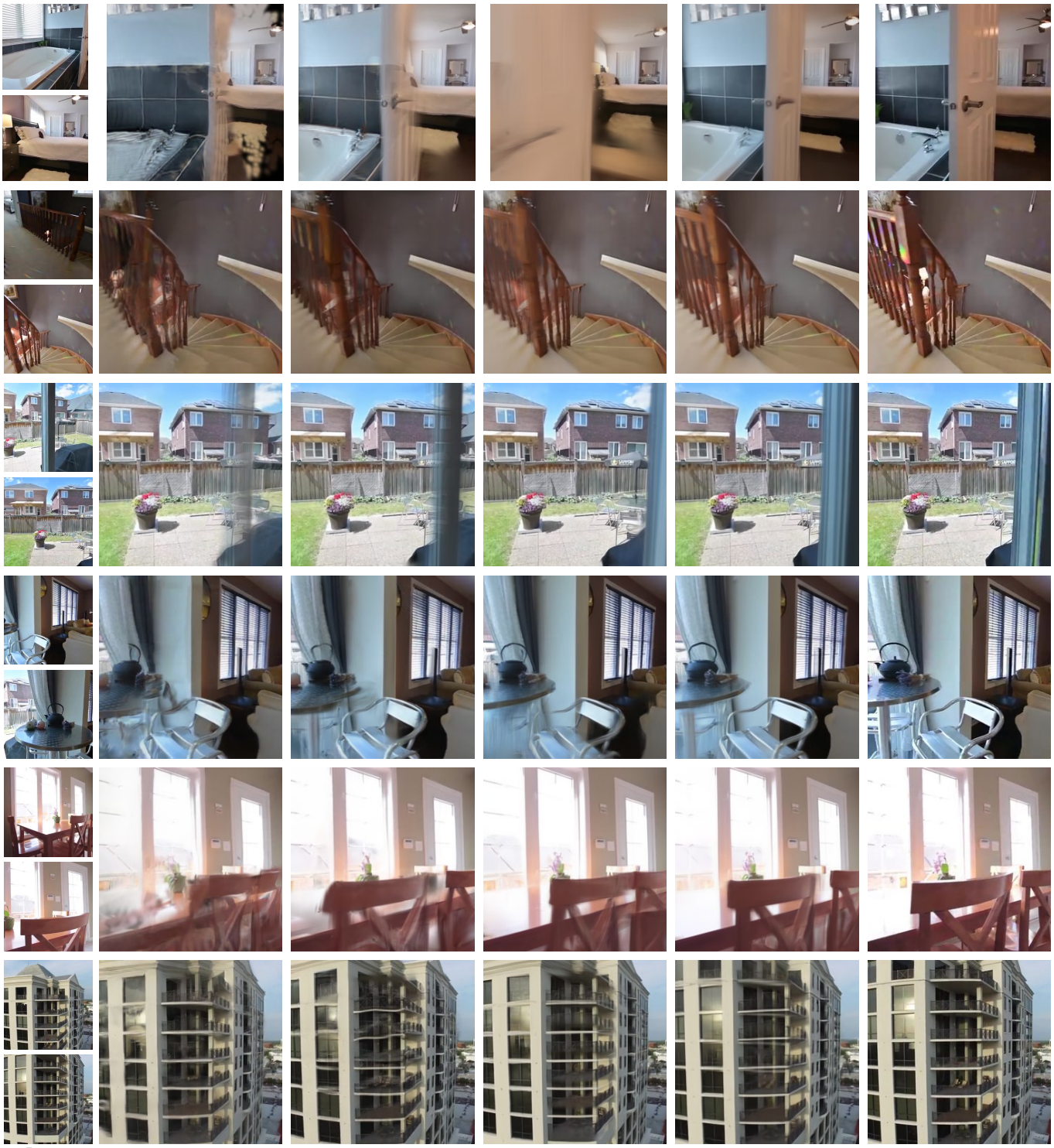

Fig. 9. **Additional qualitative comparisons on RealEstate10K at** 256 × 256 **resolution** with 2 input views.

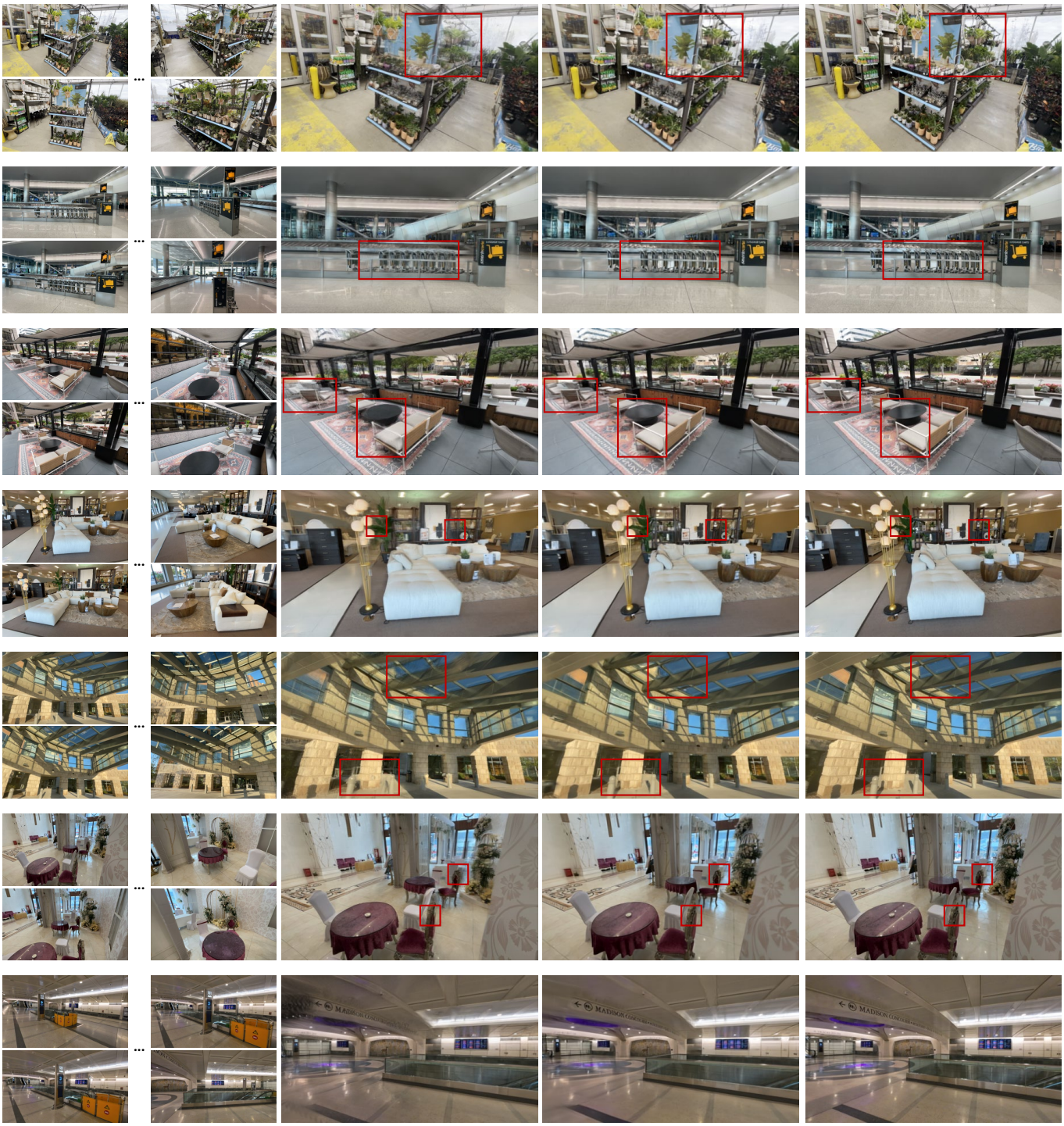

Fig. 10. **Additional Qualitative comparisons on DL3DV at** 256 × 448 **resolution** with 6 input views.

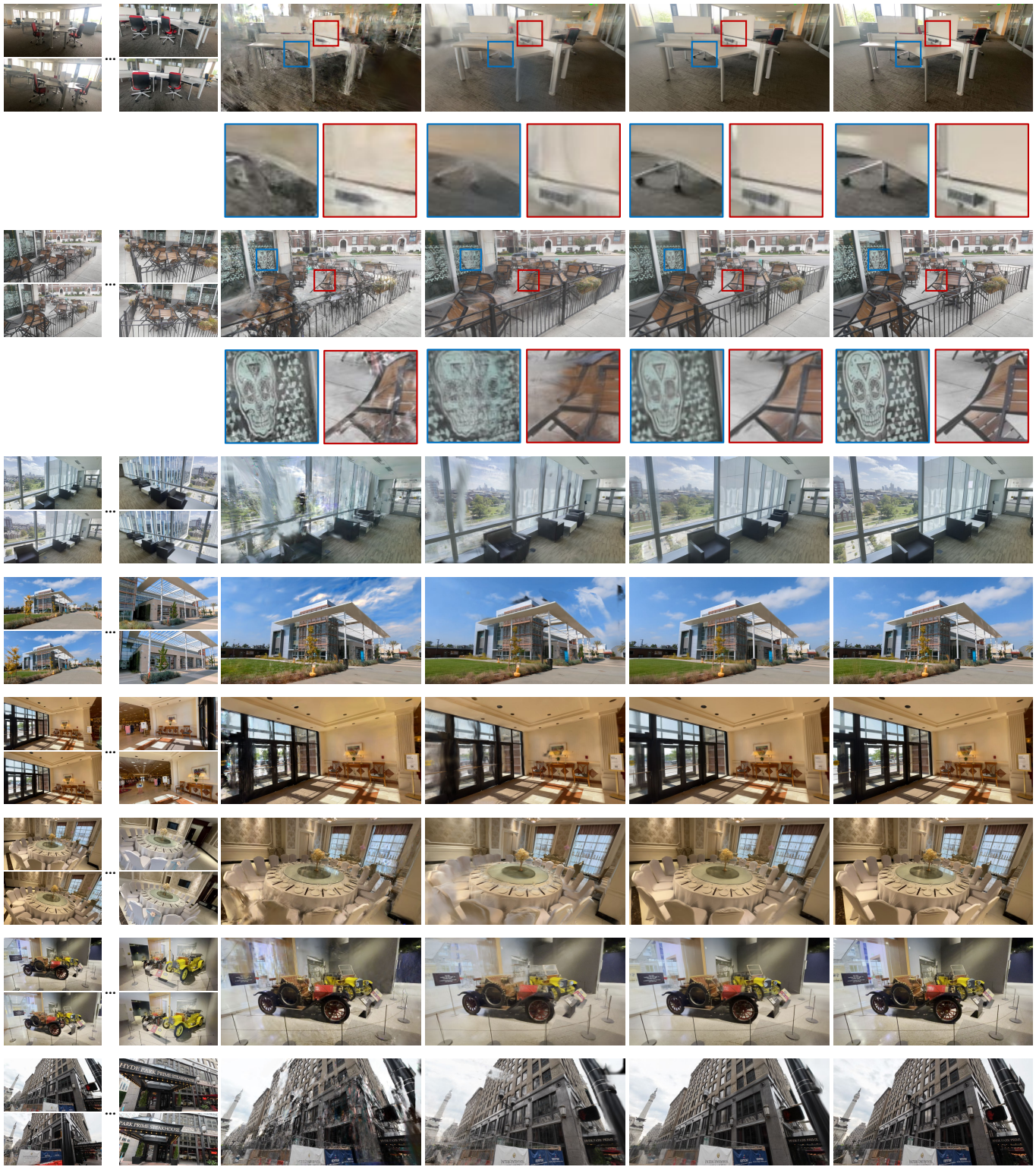

Fig. 11. **Additional Qualitative comparisons on DL3DV at** 512 × 960 **resolution** with 12 input views.