ASYSPLAT: EFFICIENT ASYMMETRIC 3D GAUSSIAN SPLATTING FOR WIDE-COVERAGE SCENE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent generalizable 3D Gaussian Splatting (3DGS) models have advanced wide-coverage novel view synthesis (NVS), but at the cost of substantial redundant computation. We identify that the redundancy can be mitigated based on two observations: (i) high-precision geometry is not strictly required for high-quality NVS; (ii) appearance learning is generally easier than geometry recovery. Motivated by these insights, we propose an asymmetric architecture that decouples geometry and appearance modeling. The geometry branch processes coarse-grained tokens with most of the parameters for multi-view reconstruction, while the appearance branch operates on fine-grained tokens to capture details using significantly fewer parameters. The two branches interact through bilateral connections, enabling mutual guidance for their respective tasks. This task-aware asymmetry reduces the computational redundancy and allocates the computation more judiciously, thereby increasing parameter efficiency and enabling smaller models to achieve strong performance. Extensive controlled experiments demonstrate that AsySplat achieves strong performance with markedly fewer parameters and reduced training and inference overhead. Combined with recent advances in generalizable 3DGS, our asymmetric design performs on par with LongLRM++ using less than 10% of its reconstruction FLOPs, and with MVP using less than 50% of its parameters and 25% of its Gaussians. Code and models will be released.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023; Yu et al., 2024; Mallick et al., 2024) has emerged as an effective representation for novel view synthesis (NVS). Recent research has shifted toward generalizable 3DGS models (Charatan et al., 2024; Chen et al., 2024c;a; Tang et al., 2024; Xu et al., 2024; Zhang et al., 2024b; Xu et al., 2025a), also referred to as Gaussian Reconstruction Models, which predict 3D Gaussians from networks. While most methods focus on sparse inputs, real-world captures often consist of wide-coverage sequences. With the rise of high-resolution devices and the growing expectations of visual detail, it becomes increasingly important to extend generalizable 3DGS methods to handle high-resolution, wide-coverage inputs (Ziwen et al., 2025).

Most current methods like LongLRM (Ziwen et al., 2025) stack layers to predict depth and Gaussian appearance attributes directly from patchified tokens. This design concentrates all computation on inter-frame token interactions. However, high-quality rendering requires small patch sizes for fine-grained modeling. At this granularity, processing 32 input views at 960P resolution produces a sequence of over 260k tokens, resulting in substantial computational overhead. This raises a fundamental question: Does such simplistic stacking introduce computational redundancy and further drive up parameter demand? In this work, we explore whether a more judicious allocation of computation can reduce computational overhead and improve parameter efficiency in NVS, enabling efficient and compact models to achieve competitive performance, as shown in Fig. 1.

In this work, we highlight two observations that are crucial for computation allocation but overlooked in prior generalizable 3DGS methods: **(i)** *High-precision geometry is not a must for high-quality rendering.* While geometry matters, the high-fidelity 3DGS rendering largely arises from the alpha compositing of Gaussians during splatting, rather than from highly precise geometry. This is consistent with reports of high rendering quality despite modest performance in surface reconstruction (Guédon & Lepetit, 2024; Huang et al., 2024), as illustrated in Fig. 1. **(ii)** *High-precision appearance attributes are necessary, while they are generally easier to model than geometry recovery.* Once multi-view depth is established, mapping RGB images to Gaussian appearance attributes is

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.

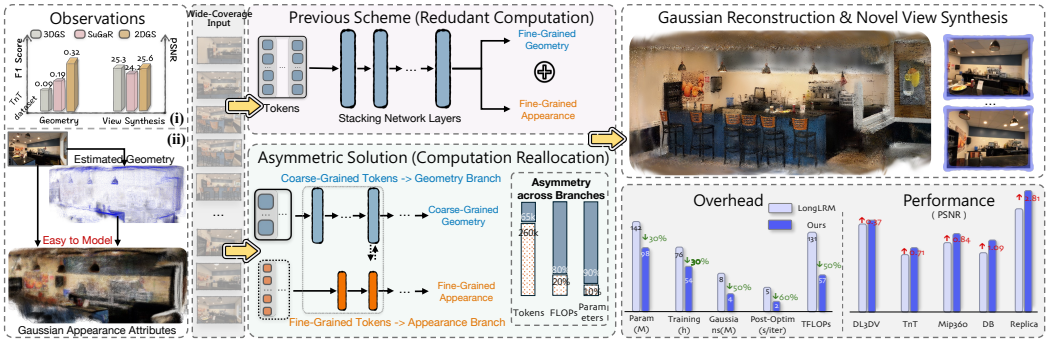

Figure 1: AsySplat reduces computational redundancy in wide-coverage Gaussian reconstruction by decoupling it into asymmetric geometry and appearance branches. The asymmetry spans token granularity, computation allocation, and parameter distribution, which is motivated by two observations: **(i)** High-precision geometry is not a must for high-quality rendering. **(ii)** Once geometry is estimated, high-precision appearance attributes are easy to model by assigning pixel-aligned RGB evidence to Gaussians. This design enables a markedly smaller model to achieve strong performance with reduced training and inference overhead.

relatively straightforward, as it mainly involves assigning pixel-aligned RGB evidence to Gaussians. This property is also explored in scene generation works (Yu et al., 2025). These insights prompt us to reconsider prevailing designs that allocate identical computation agnostically to geometry and appearance modeling (Zhang et al., 2024b; Ziwen et al., 2025).

Motivated by these insights, we propose an *Asymmetric Gaussian Splatting* (AsySplat) model that reallocates computation by decoupling the modeling of geometry and appearance into two branches, and aligning each with its precision needs and learning difficulty. This decoupled design is asymmetric, manifested in input granularity, computation, and parameter allocation. **(i)** *Geometry branch* processes coarse-grained tokens patchified with large patch sizes, reflecting the observation that high-precision geometry is not a must for NVS. Since multi-view reconstruction is intrinsically harder—requiring exhaustive inter-frame correspondence for depth estimation, we allocate the majority of parameters to this branch to ensure sufficient capacity for geometry recovery. With fewer tokens, the cost of computationally intensive inter-frame token interactions is greatly reduced. **(ii)** *Appearance branch* models the Gaussian appearance attributes from fine-grained tokens patchified with small patch sizes. Fine-grained tokens are critical for preserving details and thus necessary for high-fidelity NVS. Since appearance prediction is relatively easier, we use slimmer feature channels and far fewer parameters than in the geometry branch (roughly 10% of the total). The two branches communicate via bilateral connections: the appearance branch injects fine details into the geometry branch, while the geometry branch supplies depth cues to guide the prediction of Gaussian attributes. The asymmetric structure resolves a key dilemma of prior works, which inevitably drag fine-grained tokens into computationally expensive inter-frame token interactions. In contrast, we retain fine-grained tokens for detailed rendering while keeping them out of those heavy interactions. AsySplat not only provides a practical recipe for removing computation redundancy by allocating the computation more judiciously, but also improves per-parameter efficiency and enables smaller models to achieve strong performance.

We adopt a hybrid structure in the geometry branch to control the training overhead. However, excessive reliance on Mamba (Gu & Dao, 2024) degrades performance (Ziwen et al., 2025), since state-space models are less effective for the explicit matching required by multi-view depth estimation. We therefore reinvest the computation saved by our asymmetric design into additional attention (Vaswani et al., 2017) layers to capture contextual cues more effectively, which is crucial for compact models. However, incorporating more attention layers increases training time. To mitigate this overhead, we introduce a sparse attention module that partitions each token's context based on a 2D grid structure. This carefully designed module not only substantially reduces training time, e.g., a 40% time reduction in the computationally heaviest stage, but also maintains competitive performance.

Our contributions are summarized as follows:

- We introduce an asymmetric design to tackle computational redundancy in generalizable 3DGS for high-resolution, wide-coverage inputs. This asymmetry enables a judicious allocation of computation and parameters, leading to efficient and compact models.

- We design a sparse attention module that allows us to retain more attention layers for higher performance, while not incurring large training overhead.
- On 32-view 960P inputs, AsySplat outperforms LongLRM on every metric across five benchmarks in feed-forward reconstruction, with 30% fewer parameters, 30% less training time, over 50% fewer reconstruction FLOPs, 50% fewer Gaussians, and 60% faster post-optimization. Combined with recent advances, it matches state-of-the-art performance using less than 10% of the FLOPs, fewer than 50% of the parameters, or only 25% of Gaussians.

2 Related Work

Neural Representation for Novel View Synthesis. 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023; Yu et al., 2024; Lu et al., 2024) has emerged as a highly effective scene representation method for novel view synthesis (NVS). While Neural Radiance Fields (NeRFs) are capable of producing high-quality renderings (Mildenhall et al., 2020; Barron et al., 2021; 2022; 2023), 3DGS surpasses them not only in rendering fidelity but also in optimization efficiency and real-time rendering performance. Similar to the development of NeRF (Müller et al., 2022; Chen et al., 2022; Zhong et al., 2024; 2025b; Zhenxing & Xu, 2022; Pumarola et al., 2021; Azinović et al., 2022), subsequent 3DGS researches have extended in various directions, including reducing optimization time (Mallick et al., 2024; Chen et al., 2025b), handling sparse input views (Zhu et al., 2024b; Zhang et al., 2024a; Zhong et al., 2025a; Chen et al., 2025a), large-scale scene reconstruction (Kerbl et al., 2024; Wang & Xu, 2025), dynamic scene modeling (Wu et al., 2024; Gao et al., 2024), and improving geometry for mesh recovery (Chen et al., 2024b; Wolf et al., 2024; Huang et al., 2024).

Generalizable Scene Modeling. To circumvent the time-consuming per-scene optimization required by conventional NeRF/3DGS, recent research has explored generalizable approaches that directly predict scene representations from multi-view images in a feed-forward manner(Yu et al., 2021; Wang et al., 2021; Chen et al., 2021; Hong et al., 2023; Charatan et al., 2024; Szymanowicz et al., 2024; Chen et al., 2024c;d; Xu et al., 2025a; Zhang et al., 2024b). Following the progress of pose-dependent generalizable 3DGS, a growing line of research explores pose-free generalizable 3DGS (Ye et al., 2024; Xu et al., 2025b; Kang et al., 2025; Jiang et al., 2025). Most generalizable 3DGS methods focus on sparse views, typically using no more than 16 views. Recent works (Ziwen et al., 2025; Kang et al., 2026a) extend generalizable 3DGS to high-resolution, wide-coverage inputs, modeling scenes with up to 32 views at 960P resolution under known camera poses. We adopt this pose-given, feed-forward setting due to its practical importance for real-world NVS applications. This setting is related to recent long-sequence geometry models (Wang et al., 2025). However, those methods mainly focus on scaling to longer sequences through causal training for autoregressive rollout (Zhuo et al., 2025; Wang & Xu, 2026), or memory mechanisms (Chen et al., 2026). In contrast, our focus is not sequence scalability, but reducing computational redundancy in 3DGS-based NVS by allocating computation more rationally between geometry and appearance modeling. In this work, we advance beyond LongLRM by identifying computational redundancy within token interactions. We introduce an asymmetric architecture that mitigates this redundancy through computation reallocation, guided by the precision requirements and learning difficulty of geometric and appearance modeling for NVS.

3 Methodology

3.1 Motivation

Given wide-coverage, high-resolution inputs, most generalizable 3DGS methods patchify images into tokens and stack layers to model token interactions for the prediction of depth and appearance attributes (Ziwen et al., 2025). To meet high-fidelity NVS requirements, patches should be small to preserve detail, which imposes a heavy computational burden, since the cost of token interactions grows quadratically (Vaswani et al., 2017) or linearly (Gu & Dao, 2024) with token length. This high computational demand motivates us to explore redundancies introduced by simplistic layer stacking, which not only increases computational overhead but also escalates parameter requirements. Therefore, this work aims to reduce such redundancy through strategic computation reallocation, thus improving parameter efficiency and enabling compact models to achieve strong performance.

Our computation allocation is guided by two observations in generalizable 3DGS: **(i)** High-precision geometry is not a must for high-fidelity NVS. Studies on surface reconstruction (Guédon & Lepetit,

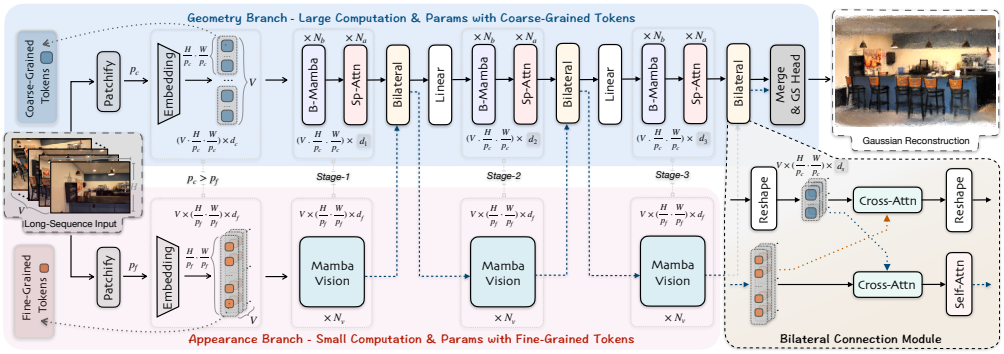

Figure 2: **Framework overview.** For Gaussian reconstruction from wide-coverage inputs, AsySplat reallocates computation by decoupling the modeling of geometry and appearance into two branches. The geometry branch processes coarse-grained tokens through three hierarchical stages of increasing capacity, using bidirectional Mamba and a sparse attention module (Fig. 3) for full *inter-frame* token interactions. The appearance branch handles fine-grained tokens with lightweight MambaVision with token interactions restricted to *intra-frame*. Bilateral connections enable cross-branch interaction. The final outputs from both branches are merged for Gaussian prediction.

2024; Huang et al., 2024) show that 3DGS can render well despite imperfect geometry, as shown in Fig. 1, largely due to alpha compositing of numerous Gaussians. This is further supported by our experiments, showing that our model achieves rendering quality comparable to state-of-the-art methods, even with geometry (Fig. 5) output from coarse-grained tokens. **(ii)** High-precision appearance attributes are necessary, while they are generally easier to model than geometry. Intuitively, once multi-view depth is established, the network only needs to map from 2D pixel-aligned cues to appearance attributes (Yu et al., 2025), without inter-frame interaction. These insights inspire us to reallocate computation by decoupling geometry and appearance into two dedicated branches, each aligned with its precision needs and learning difficulty. This design leads to the asymmetric architecture depicted in Fig. 2, with asymmetry in input granularity, computation, and parameters.

3.2 Asymmetric Gaussian Splatting Model

Given V input images $\{\mathbf{I}_i\}_{i=1}^{V}(\mathbf{I}_i \in \mathbb{R}^{H\times W\times 3})$ together with their camera intrinsics and extrinsics, our goal is to predict N 3D Gaussians (Kerbl et al., 2023) to represent the entire scene: $\{(\boldsymbol{\mu}_n, \alpha_n, \mathbf{s}_n, \mathbf{q}_n, \mathbf{SH}_n)\}_{n=1}^{N}$, corresponding to each Gaussian's center, opacity, scale, quaternion, and spherical harmonics, respectively. The asymmetry in AsySplat starts from the tokenization stage. We first concatenate the images with their associated Plücker rays, and then tokenize the inputs at two different granularities. For the geometry branch, inputs are tokenized with a large patch size p_c, yielding coarse-grained tokens $\mathbf{C}^0 \in \mathbb{R}^{(VHW/p_c^2)\times d_c}$, following observation **(i)** in Sec. 3.1. In contrast, the appearance branch uses a smaller patch size p_f, where $p_f < p_c$, to preserve details for high-fidelity rendering, producing fine-grained tokens $\mathbf{F}^0 \in \mathbb{R}^{V\times(HW/p_f^2)\times d_f}$. These two token sets are fed into the geometry and appearance branches, respectively. The subsequent asymmetric processing and fusion of these tokens are detailed below.

Geometry Branch. The geometry branch targets multi-view consistent depth recovery. Achieving this requires exhaustive matching across different views to establish correspondences. Therefore, the branch processes the entire sequence of coarse-grained tokens of length VHW/p_c^2. To balance accuracy and efficiency, the branch is structured into three hierarchical stages, with the token dimension progressively increasing through $\{d_s\}_{s=1}^{3}$. Within each stage, we adopt a hybrid architecture (Ziwen et al., 2025) that combines Mamba (Gu & Dao, 2024) with attention (Vaswani et al., 2017) layers, enabling rapid acquisition of long-range context via efficient token interactions. Given the output $\mathbf{C}^{s-1}$ from the preceding stage $s-1$ ($s \in \{1,2,3\}$), stage s performs the following operations:

$$\begin{aligned} \mathbf{B}_0^s &= \mathbf{C}^{s-1}, \mathbf{B}_l^s = \mathrm{Mamba}_l(\mathbf{B}_{l-1}^s), l = 1, ..., N_b, \\ \mathbf{T}_0^s &= \mathbf{B}_{N_b}^s, \mathbf{T}_l^s = \mathrm{SparseAttn}_l(\mathbf{T}_{l-1}^s), l = 1, ..., N_a, \end{aligned} \tag{1}$$

where N_b and N_a denote the numbers of Mamba and attention layers per stage. Both $\mathbf{B}_l^s$ and $\mathbf{T}_l^s$ have the shape $(VHW/p_c^2) \times d_s$. We adopt bidirectional Mamba (Zhu et al., 2024a), which performs

forward and backward scans to provide each token with full context, and we employ a sparse attention module (Sec. 3.3) for computational efficiency.

In summary, the geometry branch stacks layers within three hierarchical stages to recover scene geometry. At the end of each stage, we introduce a bilateral connection module that exchanges information with the appearance branch. Given the intrinsic difficulty of multi-view geometric reasoning, we allocate the majority of AsySplat's parameters to this branch. It is important to note that observation **(i)** in Sec. 3.1, stating that 'high-precision geometry is not a must', does not contradict the design of allocating more parameters to the geometry branch. This is because, even for coarse depth prediction (output from coarse-grained tokens), the network still needs to implicitly perform the challenging cross-view matching and multi-view stereo, which demands substantial network capacity. We validate the rationality of this design in our ablation studies.

Appearance Branch. In line with observation **(ii)** in Sec. 3.1, inter-frame interaction is unnecessary for estimating appearance attributes. Consequently, we restrict token interactions to be intra-frame, so the effective sequence length during computation is bounded by the per-frame token count rather than the total across V frames. Although the fine-grained token sequence $\mathbf{F}^0$ fed into the appearance branch is much longer than that of the geometry branch, its computational cost does not scale significantly with the number of views. This enables high-fidelity appearance modeling without incurring prohibitive multi-view overhead. Mirroring the geometry pathway, the appearance branch is also organized into three stages. Each stage stacks several Mamba layers to strengthen per-frame appearance modeling. Given the output $\mathbf{F}^{s-1}$ from the preceding stage, stage s applies the following operations:

$$\mathbf{M}_0^s = \mathbf{F}^{s-1}, \mathbf{M}_l^s = \mathrm{MambaVision}_l(\mathbf{M}_{l-1}^s), l = 1, ..., N_v, \tag{2}$$

where N_v denotes the number of layers per stage. $\mathbf{M}_l^s$ maintains shape of $V \times (HW/p_f^2) \times d_f$, and all operations in Eq. (2) are performed intra-frame. Unlike the hierarchical widths of the geometry branch, the appearance width d_f is kept constant across stages. We adopt MambaVision (Hatamizadeh & Kautz, 2025) in this branch, which replaces causal convolutions with standard ones and adds a non–selective-scan branch. This design enhances global modeling capacity while being more efficient than bidirectional Mamba.

At the end of each stage, the fine-grained tokens exchange information with the coarse-grained tokens via a bilateral connection module. Consistent with observation **(ii)** in Sec. 3.1 that appearance attributes are generally easier to model, we allocate substantially fewer parameters to this branch (e.g., about 10% of the total), which shares design similarities with (Feichtenhofer et al., 2019).

Bilateral Connection Module. The two branches communicate via bilateral connections. Coarse-grained tokens absorb high-frequency cues from the fine-grained tokens, while the latter receive geometric context to assist appearance attributes prediction, e.g., the scale of Gaussians is often correlated with its depth. As illustrated in Fig. 2, the communication process can be formulated as:

$$\begin{aligned} \tilde{\mathbf{C}}^s = \mathrm{Reshape}(\mathbf{T}_{N_a}^s), \quad \tilde{\mathbf{F}}^s = \mathbf{M}_{N_v}^s, \\ \mathbf{C}^s = \mathrm{Reshape}\big(\mathrm{CrossAttn}(\tilde{\mathbf{C}}^s, \tilde{\mathbf{F}}^s)\big), \quad \mathbf{F}^s = \mathrm{SelfAttn}\big(\mathrm{CrossAttn}(\tilde{\mathbf{F}}^s, \tilde{\mathbf{C}}^s)\big). \end{aligned} \tag{3}$$

where $\mathbf{T}_{N_a}^s$ and $\mathbf{M}_{N_v}^s$ are outputs from Eq. (1) and Eq. (2), respectively. The two $\mathrm{Reshape}$ operations convert tensors between shapes of $(VHW/p_c^2) \times d_s$ and $V \times (HW/p_c^2) \times d_s$, and vice versa. All computations in Eq. (3) are performed in an intra-frame manner.

Merging and Gaussian Prediction. To combine the complementary information from the two branches, we merge the two token sets to predict the final Gaussians. After reshaping them into 2D token maps of $\mathbf{C}^{\mathrm{out}}$ and $\mathbf{F}^{\mathrm{out}}$, we fuse them with a strided convolution to get an output map:

$$\mathbf{O}^{\mathrm{out}} = \mathbf{C}^{\mathrm{out}} + \mathrm{Conv}_{3\times3}(\mathbf{F}^{\mathrm{out}}, \mathrm{stride} = p_c/p_f), \tag{4}$$

whose shape is $V \times (H/p_c) \times (W/p_c) \times d_3$. Each pixel is mapped to p_f^2 Gaussians via a linear decoder. Rearranging this output yields a tensor of shape $V \times (Hp_f/p_c) \times (Wp_f/p_c) \times z$, z denotes the number of parameters per Gaussian. This tensor can be used to perform pixel-aligned predictions on images downsampled by a factor of p_c/p_f, resulting in a total of $VHW(p_f/p_c)^2$ Gaussians. For a configuration where $p_c > p_f$, the Gaussians are drastically reduced.

The asymmetry between the two branches is manifested in three aspects: input granularity, computational allocation, and parameter count. Our approach reduces computational redundancy by

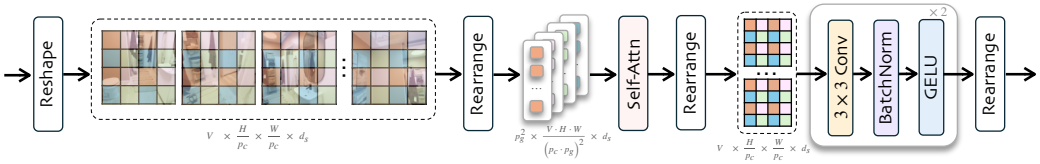

Figure 3: Illustration of the sparse attention module. The input token sequence is reshaped into 2D token maps, which are processed by two convolutional blocks to preserve contextual completeness. The refined token maps are partitioned with a uniform grid of size $p_g \times p_g$, which are rearranged into a sequence of reduced length for self-attention. p_g is set to 2 for clarity in this illustration.

performing inter-frame interactions on coarse-grained tokens while confining fine-grained tokens to intra-frame processing. This more rational computational allocation promotes specialization, where parameters are dedicated to specific tasks, thus enhancing parameter efficiency.

3.3 Sparse Attention Module

The proposed asymmetric structure reduces computational redundancy by allocating resources more judiciously. The saved computation can then be reinvested to enhance the model's capability. However, as noted by LongLRM (Ziwen et al., 2025), excessive reliance on Mamba severely degrades performance, since state-space models are less effective for the dense matching required in multi-view depth estimation. Therefore, we retain more attention layers in each stage of our geometry branch. Although the computational increase from these additional attention layers is acceptable, it leads to a substantial increase in training time. This is primarily due to the quadratic complexity with respect to the length of the coarse-grained token sequence: $\mathcal{O}((VHW/p_c^2)^2)$.

Table 1: Training overhead across stages. * replaces sparse attention with full attention.

Image Size	Method	Batch Size	Train Step	#Param	Iteration Time	GPU Memory
256×256	LongLRM	8	60k	142M	2.7s	27G
	AsySplat (Ours)		60k	98M	1.9s	25G
512×512	LongLRM	2	10k	142M	2.9s	27G
	AsySplat (Ours)		10k	98M	2.2s	25G
540×960	LongLRM	2	10k	142M	8.5s	52G
	AsySplat (Ours)		10k	98M	6.0s	48G
	AsySplat*		10k	95M	10.5s	48G

Inspired by MaxViT (Tu et al., 2022), which introduces grid attention to reduce computational overhead in single-frame processing, we extend this idea to multi-view inputs by developing a sparse attention module for inter-frame token interaction. As depicted in Fig. 3, the input token sequence is first reshaped into 2D token maps. We then apply convolutional blocks to each frame's token map independently to aggregate local information. This step is crucial for preserving rich contextual details for every token. The refined token maps are partitioned using a uniform grid of size $p_g \times p_g$, which are subsequently rearranged into a structured sequence of shape $p_g^2 \times (VHW/(p_g p_c)^2) \times d_s$, on which self-attention is performed. Finally, the self-attention output is rearranged back to the original input token sequence shape. Experiments demonstrate that applying self-attention on this reduced-length sequence significantly shortens training time, while the convolutional refinement is also essential for context integrity and thus critical to performance.

4 Experiments

Datasets. DL3DV (Ling et al., 2024) is a large-scale dataset with over 10k training scenes and 140 scenes for benchmarking. We train AsySplat on the DL3DV training split and report most results on its benchmark split. To evaluate the zero-shot generalization capability of the model, we test on four datasets, including three widely adopted 3DGS benchmarks, Tanks&Temples (Knapitsch et al., 2017), MipNeRF-360 (Barron et al., 2022), and Deep Blending (Hedman et al., 2018), as well as the synthetic Replica dataset (Straub et al., 2019).

Implementation details. We adopt a progressive training strategy with increasing resolutions of 256×256, 512×512, and 540×960, with batch sizes of 256, 64, and 64, respectively. The patch sizes p_c and p_f are set to 16 and 8. The token dimensions $\{d_s\}_{s=1}^3$ of the three geometry stages are set to $\{192, 512, 896\}$, N_v in the appearance branch is set to 2, and the grid size p_g of sparse attention is set to 2. More details are provided in the supplementary material. Unless otherwise specified, all experiments use 32 input views and are evaluated at 540×960.

Table 2: Controlled comparison with LongLRM in 32-view feed-forward reconstruction and post-optimization. '#Iter' denotes the number of post-optimization iterations initialized from the predicted Gaussians, and 'Time' denotes the total inference runtime. All benchmarks use 540×960 resolution, except Replica (544×736). Best, second, and third results are marked in red, orange, and yellow.

Method	#Iter	Time↓	DL3DV-140			Tanks&Temples			MipNeRF-360			Deep Blending			Replica		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LongLRM	-	<1s	24.10	0.783	0.254	18.38	0.601	0.363	21.54	0.529	0.417	20.91	0.701	0.464	26.12	0.869	0.201
	3	15s	24.99	0.809	0.243	18.69	0.623	0.360	22.49	0.601	0.388	21.38	0.715	0.456	27.39	0.889	0.184
	10	50s	25.60	0.826	0.233	18.90	0.642	0.350	23.13	0.632	0.373	21.84	0.725	0.450	28.63	0.899	0.176
AsySplat (Ours)	-	<1s	24.22	0.793	0.248	19.03	0.665	0.322	22.22	0.593	0.375	21.46	0.718	0.444	27.71	0.894	0.170
	3	6s	24.91	0.811	0.240	19.27	0.678	0.321	22.85	0.627	0.362	21.84	0.726	0.440	28.94	0.907	0.160
	10	18s	25.45	0.824	0.233	19.46	0.693	0.313	23.40	0.650	0.352	22.29	0.735	0.436	30.16	0.917	0.153
	20	36s	25.81	0.833	0.223	19.57	0.703	0.306	23.78	0.668	0.339	22.69	0.742	0.429	31.02	0.924	0.144
	28	50s	**25.97**	**0.838**	**0.218**	**19.61**	**0.707**	**0.301**	**23.97**	**0.676**	**0.331**	**22.93**	**0.747**	**0.424**	**31.44**	**0.927**	**0.138**

4.1 Controlled Comparison with LongLRM

Our central claim is that a more judicious allocation of computation removes redundancy, allowing a model to be efficient and compact without sacrificing quality. To verify it, we adopt LongLRM (Ziwen et al., 2025) as a clean baseline, which stacks layers in a single processing stream. Under the same training recipe and evaluation protocol, we draw three conclusions.

(i) Improved parameter efficiency. As shown in Tab. 2, AsySplat performs on par with LongLRM on the in-domain DL3DV-140 benchmark in a single forward pass (24.22 vs. 24.10 PSNR) with 30% fewer parameters (98M vs. 142M). Notably, this efficiency does not come from simply reducing parameters. As shown in Tab. A3b in the supplementary material, shrinking LongLRM to a size comparable to AsySplat noticeably degrades its performance, whereas the asymmetric allocation allows a compact model to retain the quality.

Table 3: Inference overhead. The '+post-optim' config includes a post-optimization of 10 iterations, initialized with the model's predicted Gaussians.

Image Size	Method	#Param	Time	TFLOPs	GPU Memory	#GS
540×960	LongLRM	142M	<1s	131	26G	8M
	+post-optim		50s	-	31G	
540×960	AsySplat (Ours)	98M	<1s	57	22G	4M
	+post-optim		18s	-	27G	

(ii) Reduced training and inference overhead. Tab. 1 shows that AsySplat consistently reduces iteration time and GPU memory at every training stage, cutting the estimated total training time from 76 to 54 hours. For inference, Tab. 3 shows that AsySplat reduces FLOPs by more than 50% (57T vs. 131T). It also predicts half as many Gaussians (4M vs. 8M), making post-optimization faster. Consequently, AsySplat affords more iterations under the same runtime budget. Although LongLRM is slightly ahead on DL3DV-140 at an equal number of iterations (by at most 0.15 PSNR), AsySplat_{20} (36s) surpasses LongLRM_{10} (50s) by 0.21 PSNR, and AsySplat_{28} exceeds it by 0.37 PSNR at the same 50s budget. Subscripts denote the number of post-optimization iterations.

(iii) Strong zero-shot generalization. Tab. 2 shows that, on all four zero-shot benchmarks, AsySplat outperforms LongLRM on every metric at every number of post-optimization iterations. In a single forward pass, the PSNR gains are 0.65, 0.68, 0.55, and 1.59 on Tanks&Temples, MipNeRF-360, Deep Blending, and Replica, respectively. Under the matched 50s budget, they grow to 0.71, 0.84, 1.09, and 2.81. A qualitative comparison is shown in Fig. 4.

4.2 Comparison with current methods

Wide-Coverage Input. We first compare AsySplat with recent generalizable 3DGS methods under the standard 32-view setting. Since our asymmetric design operates at the backbone level, it is orthogonal to recent advances in generalizable 3DGS, such as the decoder in LongLRM++ (Ziwen et al., 2026). Integrating this decoder into AsySplat yields a 110M model, denoted AsySplat-110. As shown in Tab. 4, it achieves competitive performance with superior efficiency, requiring less than 10% of the FLOPs of LongLRM++, fewer than 50% of the parameters of MVP (Kang et al., 2026b), or only 25% of the Gaussians of MVP. Since the code for LongLRM++ is unavailable, we estimate its FLOPs using our reimplementation, which follows the architecture described in the paper and matches its reported parameter count of 110M. AsySplat also outperforms iLRM (Kang et al., 2026a) with fewer parameters and less FLOPs. Beyond the standard 32-view setting, AsySplat-110 can directly accommodate varying numbers of input views. Its reconstruction quality improves consistently as the number of views increases from 32 to 128, indicating that the asymmetric architecture scales with input coverage, although MVP surpasses AsySplat-110 at 64 and 128 views. We hypothesize that this gap stems from the hybrid Mamba backbone rather than from the asymmetric allocation itself: as

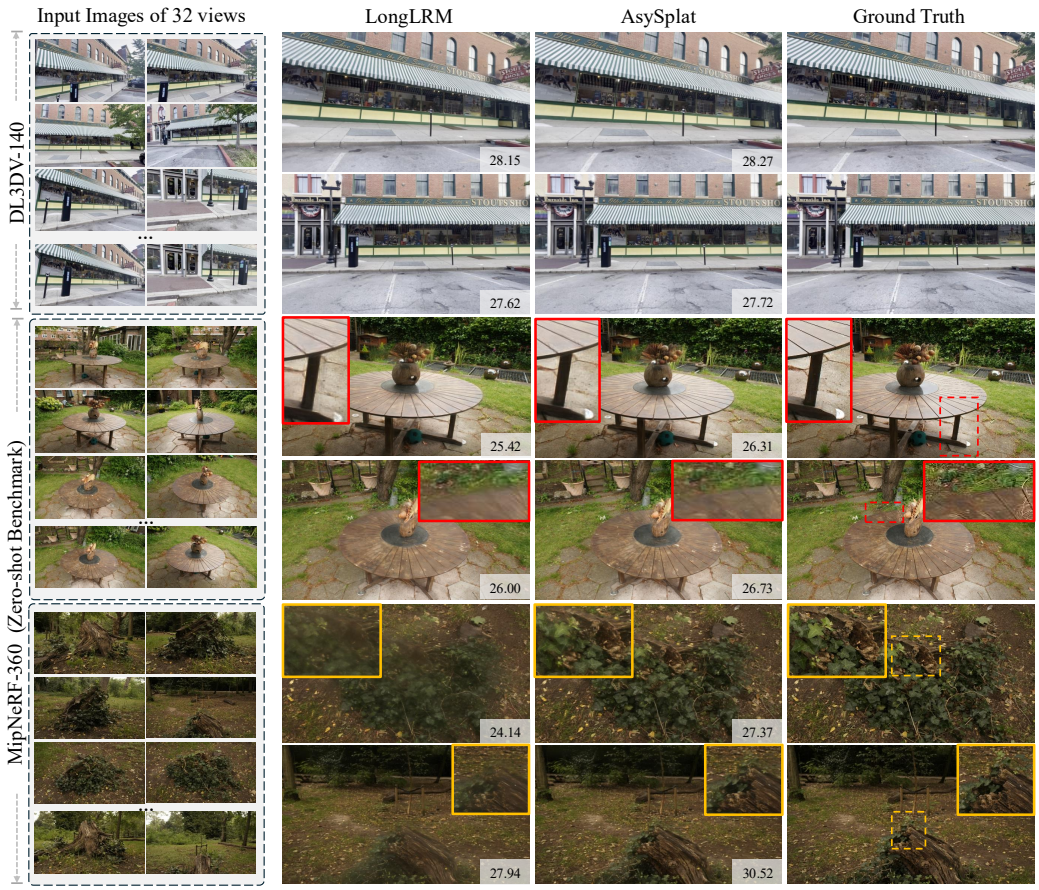

Figure 4: Qualitative comparison with LongLRM. Given 32 input views at a resolution of 540×960, both models achieve comparable quality on the DL3DV-140 benchmark, while AsySplat demonstrates superior performance on zero-shot benchmarks such as MipNeRF-360.

Table 4: Comparison on DL3DV-140 without post-optimization given different number of input views. TFLOPs and #GS are measured at 32 input views. ∼ denotes an analytic estimate.

Method	TFLOPs	#Param	#GS	16 views			32 views			64 views			128 views		
				PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LongLRM	131	142M	8M	21.05	0.708	0.297	24.10	0.783	0.254	23.60	0.789	0.260	21.24	0.739	0.308
Long-LRM++	∼700	110M	4M	–	–	–	**26.43**	0.846	0.180	–	–	–	–	–	–
iLRM	226	186M	4M	21.92	0.748	0.316	24.30	0.803	0.256	24.44	0.819	0.240	22.98	0.807	0.249
MVP	53	241M	16M	23.76	0.798	0.239	25.96	**0.847**	0.187	**27.73**	**0.881**	**0.154**	**29.02**	**0.903**	**0.134**
AsySplat-110 (Ours)	56	110M	4M	**24.82**	**0.801**	**0.213**	26.40	0.841	**0.177**	27.55	0.868	**0.154**	28.34	0.886	0.141

the coarse-token sequence grows 4× at 128 views, the limited long-range modeling of state-space layers becomes the bottleneck.

Sparse-View Input. Although our designs primarily focus on reallocating computation for wide-coverage inputs, our framework also achieves strong performance under sparse-view settings. Under the 16-view setting in Tab. 4, AsySplat-110 outperforms MVP by over 1.0 PSNR, despite using fewer than half of its parameters and only one quarter of its Gaussians. To further examine whether the proposed design remains effective under extremely sparse observations, we additionally evaluate our framework in the 2-view setting on RealEstate10K (Zhou et al., 2018), with the superior results reported in Tab. A1 of the supplementary material. Together, these results show that the proposed computation allocation remains effective across a broad range of input-view regimes.

4.3 Model Analysis

Due to resource constraints, all experiments in this section are conducted on 32 input views at a resolution of 256×256, and models are trained with a batch size of 24 for 10k iterations.

Architecture ablations. We ablate the merging operation for Gaussian prediction, the bilateral connection module for cross-branch interaction, and the MambaVision layers for appearance modeling.

Table 5: Model analysis with 32 input views at 256×256 resolution on DL3DV-140.

Method	#Param	PSNR↑	SSIM↑	LPIPS↓
AsySplat	98M	**19.24**	**0.53**	**0.49**
Pred. from fine	98M	19.02	0.52	0.50
Pred. from coarse	95M	18.54	0.47	0.54
Cross-attn w/ fine-to-coarse	89M	19.10	0.52	**0.49**
Cross-attn w/ coarse-to-fine	89M	17.33	0.44	0.54
w/o MambaVision	94M	19.06	0.51	0.51

(a) Ablations on prediction sources for Gaussians (coarse-grained or fine-grained tokens), cross-attention links in the bilateral connection module, and MambaVision in the appearance branch.

Method	#Param	PSNR↑	SSIM↑	LPIPS↓
AsySplat (4m+4s)	98M	**19.24**	**0.53**	**0.49**
7m+1s	88M	18.74	0.50	0.52
7m+1t	87M	18.78	0.50	0.52
4m+4t	95M	19.19	0.52	0.50
8t	105M	19.16	0.52	**0.49**
4m+4s (no conv)	95M	17.78	0.44	0.58

(b) Layer config per stage. m and t/s represent bidirectional Mamba and full/sparse attention. 'no conv' indicates removing convolution blocks in the sparse attention module.

	Hybrid Structure with Mamba					Alternate Attention				
	PSNR↑	SSIM↑	$\delta < 1.25$ ↑	AbsRel↓	TFLOPs↓	PSNR↑	SSIM↑	$\delta < 1.25$ ↑	AbsRel↓	TFLOPs↓
Single branch	19.1	0.52	0.32	0.69	14.8	19.3	0.53	0.41	0.52	38.3
Two branches	**19.3**	**0.53**	**0.49**	**0.44**	14.4	19.4	0.53	**0.52**	**0.45**	35.5
Asymmetric branches (Ours)	**19.3**	**0.53**	0.41	0.55	**2.9**	**19.6**	**0.55**	0.47	0.47	**4.1**

(c) Validation of the asymmetric branch design under different backbones of hybrid structure and alternate attention. The input patch sizes for the different design variants are: Single branch $p = 8$, Two branches $(p_c, p_f) = (8, 8)$, and Asymmetric branches $(p_c, p_f) = (16, 8)$.

As shown in Tab. 5a, using only coarse-grained tokens from the geometry branch or only fine-grained tokens from the appearance branch for prediction degrades performance. Removing either cross-attention link in Fig. 2 also harms performance, confirming the contribution of each direction in the bilateral design. Finally, the performance decline observed by removing MambaVision in the appearance branch confirms its contribution to appearance modeling.

Layer configuration for each stage. AsySplat processes tokens of different granularities through three stages, each composed of 8 layers. We investigate the impact of different layer configurations on performance, with results shown in Tab. 5b. Increasing the proportion of bidirectional Mamba layers ('7m+1s' or '7m+1t') leads to a performance drop, consistent with findings in LongLRM (Ziwen et al., 2025). Replacing sparse attention with full attention ('4m+4t') yields similar accuracy but substantially increases training time, as shown in Tab. 1, where iteration time rises from 6s to 10.5s. Using only full attention ('8t') further slows training without bringing accuracy gains. As discussed in Sec. 3.3, the convolution blocks in the sparse attention module are critical for context completeness, which is confirmed by the over 1.0 PSNR drop of the '4m+4s (no conv)' variant.

Design principles verification. The asymmetric design is motivated by two observations in Sec. 3.1. We verify them in Tab. 5c with rendering and geometry metrics. **(i)** Although 'Two branches' achieves more accurate geometry than 'Asymmetric branches', it does not improve NVS quality and requires higher FLOPs, supporting that *coarse geometry is sufficient*. **(ii)** Both 'Two branches' and 'Asymmetric branches' use intra-frame-only interaction in the appearance branch, whereas 'Single branch' uses inter-frame interaction for both geometry and appearance. Their stronger performance suggests that *fine appearance is easier to model and can rely on intra-frame-only interaction*. The two design principles also hold in the alternate attention structure (Wang et al., 2025).



Figure 5: AsySplat recovers the overall correct geometry from coarse-grained tokens.

5 Conclusion

This work studies computation redundancy in generalizable 3D Gaussian Splatting methods for wide-coverage, high-resolution novel view synthesis. We propose an asymmetric architecture that decouples the modeling of geometry and appearance into two branches, and allocates the computation and parameters based on their respective precision needs and learning difficulty. This design improves parameter efficiency and greatly reduces the computation redundancy.

Limitations. Although AsySplat scales consistently from 32 to 128 input views, it trails MVP at 64 and 128 views, which we attribute to the limited long-range modeling of the hybrid Mamba backbone rather than to the asymmetric allocation itself. Combining our design with a backbone better suited to very long sequences is left for future work.

Reproducibility statement

To facilitate reproducibility, we provide the architecture hyperparameters, training procedure, loss functions and weights, evaluation protocol, post-optimization settings, and overhead measurement settings in Sec. 4 and the supplementary material. All datasets used in this work are publicly available. Code and models will be released upon publication.

AI use statement

In this work, we used generative AI tools to polish the writing of the paper and to assist in running evaluation scripts. All research ideas originate entirely from the authors. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *CVPR*, 2022.

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024.

Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022.

Anpei Chen, Haofei Xu, Stefano Esposito, Siyu Tang, and Andreas Geiger. Lara: Efficient large-baseline radiance fields. In *ECCV*, 2024a.

Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 2024b.

Kangjie Chen, Yingji Zhong, Zhihao Li, Jiaqi Lin, Youyu Chen, Minghan Qin, and Haoqian Wang. Quantifying and alleviating co-adaptation in sparse-view 3d gaussian splatting. *arXiv preprint arXiv:2508.12720*, 2025a.

Lin-Zhuo Chen, Jian Gao, Yihang Chen, Ka Leong Cheng, Yipengjing Sun, Liangxiao Hu, Nan Xue, Xing Zhu, Yujun Shen, Yao Yao, and Yinghao Xu. Geometric context transformer for streaming 3d reconstruction. *arXiv preprint arXiv:2604.14141*, 2026.

Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

Youyu Chen, Junjun Jiang, Kui Jiang, Xiao Tang, Zhihao Li, Xianming Liu, and Yinyu Nie. Dash-gaussian: Optimizing 3d gaussian splatting in 200 seconds. In *CVPR*, 2025b.

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 *ECCV*, 2024c.

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 *NeurIPS*, 2024d.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019.

Quankai Gao, Qiangeng Xu, Zhe Cao, Ben Mildenhall, Wenchao Ma, Le Chen, Danhang Tang, and Ulrich Neumann. Gaussianflow: Splatting gaussian dynamics for 4d content creation. *arXiv preprint arXiv:2403.12365*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *CVPR*, 2024.

Ali Hatamizadeh and Jan Kautz. Mambavision: A hybrid mamba-transformer vision backbone. In *CVPR*, 2025.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

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

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *arXiv preprint arXiv:2505.23716*, 2025.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

Gyeongjin Kang, Jisang Yoo, Jihyeon Park, Seungtae Nam, Hyeonsoo Im, Sangheon Shin, Sangpil Kim, and Eunbyung Park. Selfsplat: Pose-free and 3d prior-free generalizable 3d gaussian splatting. In *CVPR*, 2025.

Gyeongjin Kang, Seungtae Nam, Seungkwon Yang, Xiangyu Sun, Sameh Khamis, Abdelrahman Mohamed, and Eunbyung Park. ilrm: An iterative large 3d reconstruction model. In *CVPR*, 2026a.

Gyeongjin Kang, Seungkwon Yang, Seungtae Nam, Younggeun Lee, Jungwoo Kim, and Eunbyung Park. Multi-view pyramid transformer: Look coarser to see broader. In *CVPR*, 2026b.

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

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024.

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

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *CVPR*, 2024.

Ilya Loshchilov and Frank Hutter. 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. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *CVPR*, 2024.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Francisco Vicente Carrasco, Markus Steinberger, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. *arXiv preprint arXiv:2406.15643*, 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 *ECCV*, 2020.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4): 1–15, 2022.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021.

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *CVPR*, 2024.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *ECCV*, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *ECCV*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021.

Zipeng Wang and Dan Xu. Hyrf: Hybrid radiance fields for memory-efficient and high-quality novel view synthesis. *arXiv preprint arXiv:2509.17083*, 2025.

Zipeng Wang and Dan Xu. Flashvggt: Efficient and scalable visual geometry transformers with compressed descriptor attention. In *CVPR*, 2026.

Yaniv Wolf, Amit Bracha, and Ron Kimmel. Gs2mesh: Surface reconstruction from gaussian splatting via novel stereo views. In *ECCV*, 2024.

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 *CVPR*, 2024.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *CVPR*, 2025a.

Jiale Xu, Shenghua Gao, and Ying Shan. Freesplatter: Pose-free gaussian splatting for sparse-view 3d reconstruction. In *ICCV*, 2025b.

Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. In *ECCV*, 2024.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *NeurIPS*, 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. *arXiv preprint arXiv:2410.24207*, 2024.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021.

Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T Freeman, and Jiajun Wu. Wonderworld: Interactive 3d scene generation from a single image. In *CVPR*, 2025.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *CVPR*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019.

Jiawei Zhang, Jiahe Li, Xiaohan Yu, Lei Huang, Lin Gu, Jin Zheng, and Xiao Bai. Cor-gs: sparse-view 3d gaussian splatting via co-regularization. In *ECCV*, 2024a.

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 *ECCV*, 2024b.

MI Zhenxing and Dan Xu. Switch-nerf: Learning scene decomposition with mixture of experts for large-scale neural radiance fields. In *ICLR*, 2022.

Yingji Zhong, Lanqing Hong, Zhenguo Li, and Dan Xu. Cvt-xrf: Contrastive in-voxel transformer for 3d consistent radiance fields from sparse inputs. In *CVPR*, 2024.

Yingji Zhong, Zhihao Li, Dave Zhenyu Chen, Lanqing Hong, and Dan Xu. Taming video diffusion prior with scene-grounding guidance for 3d gaussian splatting from sparse inputs. In *CVPR*, 2025a.

Yingji Zhong, Kaichen Zhou, Zhihao Li, Lanqing Hong, Zhenguo Li, and Dan Xu. Empowering sparse-input neural radiance fields with dual-level semantic guidance from dense novel views. *arXiv preprint arXiv:2503.02230*, 2025b.

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

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024a.

Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *ECCV*, 2024b.

Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4d visual geometry transformer. *arXiv preprint arXiv:2507.11539*, 2025.

Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *ICCV*, 2025.

Chen Ziwen, Hao Tan, Peng Wang, Zexiang Xu, and Li Fuxin. Long-lrm++: Preserving fine details in feed-forward wide-coverage reconstruction. In *CVPR*, 2026.

ASYSPLAT: EFFICIENT ASYMMETRIC 3D GAUSSIAN SPLATTING FOR WIDE-COVERAGE SCENE MODELING

SUPPLEMENTARY MATERIAL

A IMPLEMENTATION DETAILS

Architecture. The patch sizes for tokenization, i.e., p_c and p_f are set to 16 and 8, respectively. The initial embedding dimensions d_c and d_f are set to 192 and 256, respectively. In the geometry branch, the token dimensions follow a hierarchical design across three stages. We set $\{d_s\}_{s=1}^{3}$ to $\{192, 512, 896\}$. In each stage, N_b and N_a are both set to 4. The expansion ratios in the sparse attention layers are set to 1, 2, and 3 for the three successive stages, respectively. In the appearance branch, we set N_v to 2 and keep d_f fixed at 256 across stages. For the sparse attention layers, we set the uniform grid size p_g to 2.

All transformer blocks in AsySplat, including the sparse attention module in the geometry branch and the full attention layers in the bilateral connection module, apply RMSNorm (Zhang & Sennrich, 2019) after the query and key projections (Touvron et al., 2023). The geometry branch utilizes bidirectional Mamba (Zhu et al., 2024a) layers with a state dimension of 256, while the appearance branch employs MambaVision (Hatamizadeh & Kautz, 2025) layers with a state dimension of 16. For Gaussian attribute prediction, we set the degree of spherical harmonics to 3.

Training. For the controlled comparison with LongLRM, We employ a progressive training strategy, increasing the resolution sequentially: 256×256, 512×512, and 540×960. The model is trained using the AdamW optimizer (Loshchilov & Hutter, 2017) with a weight decay of 0.05 across all stages. The peak learning rates for the three stages are set to 4e-4, 4e-5, and 4e-5, respectively, and are decayed following a cosine schedule. Additionally, a linear warmup of the learning rate is applied for the first 2k iterations in the initial stage. We use 32 H800 GPUs to train the AsySplat. AsySplat-110 keeps the same optimizer, weight decay and cosine schedule, but trains with 8 input views for the first three stages (256×256, 512×512 and 540x960) and raises the count to 32 in a fourth stage at 540×960, rather than using 32 views throughout. The first stage runs 60k iterations at a peak learning rate of 4e-4 with a 2k-iteration warmup; each later stage runs 10k at 4e-5 without warmup. Following MVP (Kang et al., 2026b), we further introduce a fifth mixed-input-view training stage at 540×960, in which the number of input views is sampled from $16, 32, 64, 128$. This stage runs for 20k iterations at 4e-5 without warmup.

Training overhead. In the paper, for overhead comparisons with LongLRM (Ziwen et al., 2025) at 256×256 and 512×512 resolutions, we report overhead based on randomly initialized parameters, averaging the first 20 iterations due to the unavailability of LongLRM checkpoints for these resolutions. Generally speaking, the computational overhead of LongLRM increases more significantly than that of AsySplat as training progresses. This is because predicting 4× Gaussians leads to a steeper rise in GPU memory consumption and iteration time. For the overhead at 540×960, we use the officially released LongLRM checkpoint. We compare it with our final model by continuing training for 20 iterations and report the average overhead. The training overhead is measured on H800 GPUs.

Inference overhead. We compare the inference overhead of AsySplat with LongLRM at a resolution of 540×960. The evaluation metrics reported in the main paper include inference time, TFLOPs, GPU memory usage, and the number of Gaussians. For inference time, we compute the average over the DL3DV-140 benchmark (Ling et al., 2024) after excluding the longest time as an outlier. For TFLOPs, we measure the computational cost of the network using the `thops` library. This measurement does not include the computation in the splatting. Note that at 540×960, LongLRM initially predicts 4× as many Gaussians as our method, but then prunes them from 16M to 8M based on predicted opacities. The inference overhead is measured on H800 GPUs.

Evaluation settings. We follow LongLRM to determine the input and test splits for all evaluations, whether on the DL3DV-140 benchmark or other zero-shot datasets (Knapitsch et al., 2017; Hedman et al., 2018; Barron et al., 2022; Straub et al., 2019). For a given set of scene images, we uniformly sample every 8th image in the sequence as the test split. The input views are selected from the remaining images using K-means clustering based on camera positions and viewing directions, aiming to achieve optimal scene coverage. The number of clusters is set equal to the number of input views, and the cameras closest to the cluster centers are chosen as the input split. In the ablation experiments conducted on DL3DV-140 at 256×256 resolution, we use the first 96 frames of each video sequence. Every 8th frame is uniformly selected as the test split, and 32 input views are uniformly sampled from the remaining frames. Regarding the comparisons on the Replica dataset, we select scenes following Zhong et al. (2025b). Note that Replica is the only benchmark evaluated at 544×736 resolution rather than 540×960.

Post optimization. To ensure a fair comparison with LongLRM, we apply an identical post-optimization procedure using same hyperparameters. Specifically, we render the input views from the predicted Gaussians and compute the MSE loss between the rendered and ground truth images. The loss is then backpropagated to update the Gaussian parameters. The learning rates for the center and spherical harmonics are set to 5e-4 and 1e-3, respectively, while those for opacity, scale, and quaternion are set to 0.

Training objective. To supervise AsySplat, we render predicted Gaussians from V_t target poses, obtaining $\{\hat{\mathbf{I}}_i\}_{i=1}^{V_t}$, which are compared against the corresponding ground truth images $\{\mathbf{I}_i\}_{i=1}^{V_t}$. Following prior work (Zhang et al., 2024b; Ziwen et al., 2025), we use an MSE term and a perceptual loss (Johnson et al., 2016; Chen & Koltun, 2017): $\mathcal{L}_{\text{image}} = \frac{1}{V_t}\sum_{i=1}^{V_t} \text{MSE}(\hat{\mathbf{I}}_i, \mathbf{I}_i) + \lambda_p \cdot \text{Perceptual}(\hat{\mathbf{I}}_i, \mathbf{I}_i)$, where λ_p is a weighting factor. Following LongLRM (Ziwen et al., 2025), we also adopt geometry supervision from monocular disparity estimates (Yang et al., 2024) for training stability: $\mathcal{L}_{\text{depth}} = \frac{1}{V_t}\sum_{i=1}^{V_t} \text{SmoothL1}(\hat{\mathbf{D}}_i, \mathbf{D}_i)$, where $\hat{\mathbf{D}}_i$ and $\mathbf{D}_i$ denote the normalized predicted and monocular-estimated disparity maps. Both maps are downsampled by a factor of p_c/p_f. Therefore, the overall objective is:

$$\mathcal{L} = \mathcal{L}_{\text{image}} + \lambda_d \cdot \mathcal{L}_{\text{depth}}, \tag{5}$$

where λ_d balances the depth supervision. Note that although we incorporate monocular depth supervision, AsySplat still models coarse-grained geometry due to the coarse-grained input tokens. Moreover, the downsampled output $\hat{\mathbf{D}}_i$ provides an even coarser representation of the overall scene geometry, since the scene geometry is modeled by pixel-aligned prediction. The weighting factors λ_p and λ_d are set to 0.5 and 0.01. Note that AsySplat-110 is supervised by photometric losses only, with no depth supervision.

B More Comparisons

Comparisons with state-of-the-art methods. We provide comprehensive comparisons with recent generalizable 3DGS methods in Tab. A2. The results further demonstrate that our method achieves competitive performance in both in-distribution and zero-shot settings while offering substantial efficiency advantages. Compared with MVP (Kang et al., 2026b), AsySplat-110 uses fewer than half of the parameters and only one quarter of the Gaussians, while achieving competitive rendering quality. The results on Tanks&Temples, MipNeRF-360, Deep Blending, and Replica further show that the effectiveness of our asymmetric design is not restricted to the training distribution. The base AsySplat model also consistently improves over LongLRM (Ziwen et al., 2025) across these zero-shot benchmarks, despite using fewer parameters and Gaussians. Although our method is primarily designed for high-resolution, wide-coverage reconstruction, Tab. A1 additionally shows that the same architecture remains effective in the complementary sparse-view regime. With only two input views on RealEstate10K (Zhou et al., 2018), AsySplat achieves the best PSNR and LPIPS among the compared methods using a relatively compact 98M model.

Table A1: Comparison with generalizable 3DGS methods under the 2-view setting on RealEstate10K.

Method	#Param	PSNR↑	SSIM↑	LPIPS↓
pixelSplat	125M	25.89	0.858	0.142
MVSplat	12M	26.39	0.869	0.128
GS-LRM	300M	28.10	**0.892**	0.114
DepthSplat	354M	27.47	0.889	0.114
AsySplat (Ours)	98M	**28.27**	**0.892**	**0.105**

Table A2: Comparison using 32 input views without post-optimization. $\sim$ denotes an analytical estimate. Except for Replica, which is evaluated at 544×736 resolution, the other four benchmarks are evaluated at 540×960 resolution. TFLOPs and #GS are measured at 540×960 resolution. Except for DL3DV results taken directly from the corresponding papers, results from other works are evaluated using their released models. The best, second-best, and third-best results are highlighted in red, orange, and yellow, respectively.

Method	TFLOPs	#Param	#GS	DL3DV-140			Tanks&Temples			MipNeRF-360			Deep Blending			Replica		
				PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
iLRM	226	186M	4M	24.30	0.803	0.257	18.58	0.616	0.384	21.29	0.502	0.462	21.94	0.718	0.486	**29.41**	**0.905**	0.186
MVP	53	241M	16M	25.96	**0.847**	0.187	19.54	0.701	0.277	**23.85**	**0.688**	**0.285**	22.11	**0.747**	0.418	28.40	0.900	0.140
LongLRM	131	142M	8M	24.10	0.783	0.254	18.38	0.601	0.363	21.54	0.529	0.417	20.91	0.701	0.464	26.12	0.869	0.201
LongLRM++	~700	110M	4M	**26.43**	0.846	0.180	-	–	–	–	–	–	–	–	–	–	–	–
AsySplat (Ours)	56	98M	4M	24.22	0.793	0.248	19.03	0.665	0.322	22.22	0.593	0.375	21.46	0.718	0.444	27.71	0.894	0.170
AsySplat-110 (Ours)	56	110M	4M	26.40	0.841	**0.177**	**20.42**	**0.708**	**0.251**	23.43	0.620	0.315	**22.17**	0.746	**0.369**	28.78	0.898	**0.137**

C More Ablations

Ablations on Mamba ratio. In the main paper, we claim that 'excessive reliance on Mamba degrades performance, since state-space models are less effective for the explicit matching required by multi-view depth estimation.' To substantiate this claim, we conduct experiments in this section by varying the Mamba ratio at each stage while keeping the overall parameter count similar. The results are presented in Tab. A3c. For novel view synthesis, we observe a sweet spot at 2m6a, beyond which increasing the Mamba ratio (from 2m6a to 8m0a) leads to a monotonic decline in NVS performance, consistent with our assertion that '*excessive reliance* on Mamba degrades performance'. To further demonstrate that this degradation is linked to Mamba's ineffectiveness in explicit matching, we also report depth estimation metrics, as depth accuracy serves as a reliable indicator of matching quality. As the Mamba ratio increases beyond 2m6a, depth accuracy declines, following a trend similar to that of the NVS metrics. The deterioration in geometric accuracy strongly suggests that *excessive reliance* on Mamba impairs the model's cross-view matching capability within the geometry branch.

Models with similar parameters. This work explores redundancy reduction through computational reallocation, enabling compact models to achieve strong performance. To demonstrate that designing an effective compact model is non-trivial, we compare AsySplat (98M, with a 24-layer geometry branch and a 6-layer appearance branch) against several parameter-reduced variants of LongLRM (Ziwen et al., 2025) (the official 142M, 24-layer model). These include a shallower 18-layer model, a 24-layer model with reduced feature dimensions, and a model using the same hierarchical parameters across three stages as AsySplat. Tab. A3b shows that AsySplat outperforms all these variants, confirming that its more rational computation allocation dedicates parameters to specific tasks more effectively, achieving high performance with improved parameter efficiency.

Parameter allocation ablations. Our method adopts an asymmetric structure. In this part, we ablate the parameter ratio between the geometry branch and the appearance branch. We vary the proportions of the branches while keeping the total parameter count roughly unchanged. The experimental results are shown in Tab. A3a. The results indicate that allocating 90% of the parameters to the geometry branch achieves a decent performance, across benchmarks.

Structural ablations at high resolution. To verify that the structural ablations in the main paper (Tab. 5a) hold at the target resolution, we fine-tune each variant from its 256×256 checkpoint to 540×960 and report the results in Tab. A3d. Although the absolute numbers are lower than those of our fully trained model due to the short schedule, the trends are consistent: every variant underperforms the full model.

Table A3: Model analysis with 32 input views. Unless otherwise specified, all experiments are conducted on DL3DV-140 at 256×256 resolution.

Geometry Branch Ratio	#Param	DL3DV-140 PSNR↑	DL3DV-140 SSIM↑	MipNeRF-360 PSNR↑	MipNeRF-360 SSIM↑
90%	98M	**19.24**	**0.53**	**17.52**	0.30
80%	96M	19.16	0.52	17.30	**0.31**
70%	98M	19.20	0.52	16.34	0.27
60%	98M	19.22	**0.53**	16.65	0.27
50%	96M	18.97	0.51	16.50	0.27

(a) Ablations on the parameter allocation between the geometry and appearance branches, with similar amount of parameters.

Method	#Param	PSNR↑	SSIM↑	LPIPS↓
AsySplat-30	98M	**19.24**	**0.53**	**0.49**
LongLRM-18	103M	18.39	0.48	0.53
LongLRM-24	99M	18.39	0.48	0.53
LongLRM-24#	102M	17.71	0.43	0.59

(b) Comparison of models with similar parameters. The number suffixed to each model indicates its total number of layers, and # denotes the use of hierarchical parameters across stages.

#Mamba	#Attn	Abbrev.	Param (M)	DL3DV-140 PSNR↑	DL3DV-140 SSIM↑	Mip-NeRF360 PSNR↑	Mip-NeRF360 SSIM↑	ScanNet++ AbsRel↓	ScanNet++ δ<1.25↑
8	0	8m0a	93.6	18.24	0.47	16.65	0.30	0.64	0.21
7	1	7m1a	96.6	18.80	0.49	16.91	0.29	0.57	0.35
6	2	6m2a	99.6	19.04	0.51	16.92	0.29	0.55	0.36
4	4	4m4a	95.0	19.19	**0.52**	17.06	0.30	0.44	0.39
2	6	2m6a	99.9	**19.30**	**0.52**	**17.49**	**0.32**	**0.39**	**0.44**
0	8	0m8a	105.0	19.16	**0.52**	17.42	0.29	0.43	0.40

(c) Ablations on the Mamba ratio in each stage (8 layers per stage, no sparse attention). Depth metrics on ScanNet++ indicate the capability of cross-view matching.

Method	PSNR↑	SSIM↑
AsySplat (Ours)	**17.71**	**0.54**
Pred. from fine	17.22	0.52
Pred. from coarse	17.16	0.52
Cross-attn w/ fine-to-coarse	17.37	0.53
Cross-attn w/ coarse-to-fine	17.45	0.53
w/o MambaVision	17.33	0.52
w/o Cross-attn	16.40	0.50

(d) Ablations on structural components at 540×960 resolution.