iLRM: An Iterative Large 3D Reconstruction Model

Gyeongjin Kang[1] **Seungtae Nam**[2] **Xiangyu Sun**[1] **Sameh Khamis**[3]
Abdelrahman Mohamed[3] **Eunbyung Park**[2*]

[1]Sungkyunkwan University [2]Yonsei Universiy [3]Rembrand

Project Page

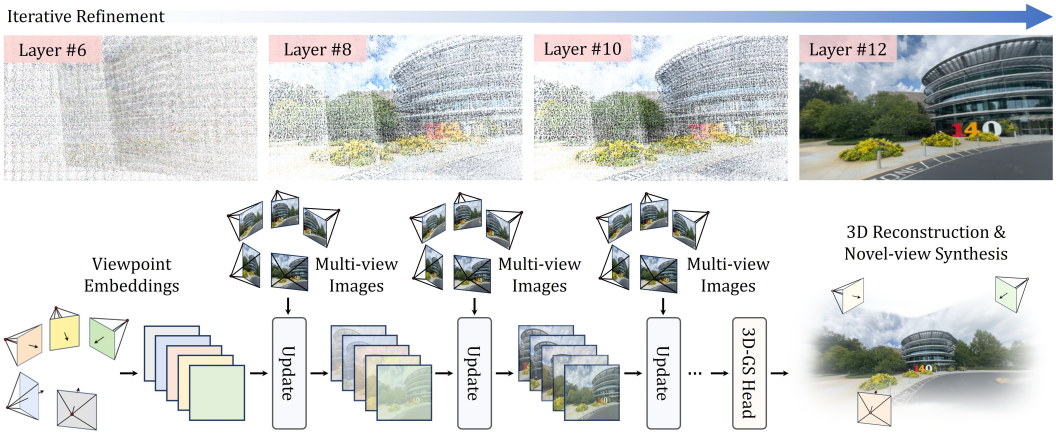


Figure 1: The overall architecture and qualitative results of the proposed *iLRM*.

Abstract

Feed-forward 3D modeling has emerged as a promising approach for rapid and high-quality 3D reconstruction. In particular, directly generating explicit 3D representations, such as 3D Gaussian splatting, has attracted significant attention due to its fast and high-quality rendering, as well as numerous applications. However, many state-of-the-art methods, primarily based on transformer architectures, suffer from severe scalability issues because they rely on full attention across image tokens from multiple input views, resulting in prohibitive computational costs as the number of views or image resolution increases. Toward a scalable and efficient feed-forward 3D reconstruction, we introduce an iterative Large 3D Reconstruction Model (*iLRM*) that generates 3D Gaussian representations through an iterative refinement mechanism, guided by three core principles: (1) decoupling the scene representation from input-view images to enable *compact 3D representations*; (2) decomposing fully-attentional multi-view interactions into a *two-stage attention* scheme to reduce computational costs; and (3) injecting *high-resolution information at every layer* to achieve high-fidelity reconstruction. Experimental results on widely used datasets, such as RE10K and DL3DV, demonstrate that *iLRM* outperforms existing methods in both reconstruction quality and speed. Notably, *iLRM* exhibits superior scalability, delivering significantly higher reconstruction quality under comparable computational cost by efficiently leveraging a larger number of input views.

*Corresponding author.

Preprint. Under review.

1 Introduction

Since the recent success of 3D Gaussian Splatting (3D-GS) [29], significant progress has been made in leveraging this 3D representation for building generalizable feed-forward 3D reconstruction models [5, 48, 54, 11, 12, 53, 59]. These methods typically train large neural networks to transform multi-view input images into feature representations, then regress Gaussian attributes. In contrast to per-scene 3D-GS optimization approaches [29, 38, 37, 19], these feed-forward models can reconstruct 3D scenes in a single forward pass, offering near real-time performance. Moreover, the prior knowledge learned from large-scale datasets [61, 33, 16, 15] allows them to effectively generalize to unseen scenes. While their reconstruction quality often lags behind that of per-scene optimization methods, fast reconstruction speed and generalization capability mark a promising step toward the long-standing goal of achieving accurate and real-time 3D scene reconstruction.

Among the promising approaches, pixel-aligned Gaussian models [5, 47, 60] have emerged as the de facto standard, leveraging decades of advances in network architectures developed for numerous image-based tasks. While these models have proven effective, they also exhibit certain limitations. In particular, since they generate per-pixel Gaussians directly from the input images, the image resolution determines the number of Gaussians produced, which can lead to an excessive number of redundant Gaussians. For example, given input images at 1K resolution across 200 viewpoints (a scale comparable to the bicycle scene in the mip-NeRF 360 dataset [2]), these methods would produce 200 million Gaussians, despite prior studies [31, 19, 18, 10, 30] demonstrating that such scenes can be efficiently represented with around 0.5 million Gaussians. To mitigate this issue, several techniques have been proposed, such as Gaussian regularization [62] and feature fusion [50]. Alternatively, the network architecture can also be designed to generate fewer Gaussians, for example by downsampling the output resolutions. However, these strategies still require processing high-resolution multi-view images and therefore do not address another fundamental limitation of these models: high computational and memory demands.

A significant portion of computational and memory overhead arises from modeling interactions across multiple input views in feed-forward 3D reconstruction models. For instance, GS-LRM [59] performs full attention over all image tokens from every input view, leading to a quadratic increase in complexity with respect to both the number of views and image resolution. MVSplat [11] and DepthSplat [53] construct and process dense cost volumes for each view, further contributing to the overall computational demands. While one might attempt to alleviate this burden by reducing the input image resolution or using a sparser set of views, such strategies risk discarding essential geometric and appearance information required for accurate reconstruction.

Beyond the computational complexity and the inefficiency of the generated representations, we also question whether the prevailing formulation, casting 3D reconstruction as a sequence-to-sequence problem that maps entire sets of image tokens to dense, pixel-aligned Gaussians, is fundamentally well-suited to the nature of the task. While this formulation has achieved impressive results (even without explicit 3D inductive biases [59, 56]), it remains primarily a one-shot 3D scene generation process. In contrast, the recent optimization-based methods [29, 38] follow a fundamentally different strategy: they treat reconstruction as an iterative refinement process, where each iteration involves rendering the current scene estimate, measuring reconstruction error, and updating the representation accordingly. This loop enables the model to progressively capture finer geometric and appearance details while ensuring strong 3D consistency. The success of these methods suggests that high-quality 3D reconstruction may benefit not only from expressive representations but also from feedback-driven iterative refinement, a trait largely absent in existing feed-forward 3D models.

In this paper, we introduce *iLRM*, an iterative large 3D reconstruction model that effectively 1) incorporates the principles of feedback-driven refinement, while also 2) addressing the computational burden and representational inefficiencies inherent in existing feed-forward approaches. As illustrated in Fig. 1, the network (acting as an optimizer) transforms the embedding features (analogous to updating the 3D-GS representation) at each layer (analogous to each optimization step), based on multi-view image tokens (serving as gradient-like signals). This design allows the model to iteratively update the scene representation at every layer based on feedback from the input images, effectively mimicking the optimization process within a feed-forward architecture. Through this process, the learned neural network jointly examines the input view images and the evolving scene representation to identify where and how to make targeted updates that improve reconstruction quality.

Another core design principle of our approach is to decouple the representation, later transformed into 3D Gaussians, from direct dependence on input images, addressing the computational complexity and redundancy that arise in architectures that generate pixel-aligned Gaussians directly from multi-view inputs. By decoupling the representation and the input images, we can use low-resolution representations to produce a compact set of Gaussians while still leveraging high-resolution input images for detailed guidance.

In addition, we propose an efficient mechanism for modeling the interaction between the representations and the input images. A naïve approach would involve computing full attention between all tokens across views, which quickly becomes computationally prohibitive. To overcome this, we initialize our representation using viewpoint embeddings, each tied to a specific input view. Interaction modeling is then split into two stages. First, we perform cross-attention between each viewpoint embedding and its corresponding image, which is highly efficient due to the one-to-one mapping. Next, we apply self-attention across all viewpoint embeddings. Importantly, since this second stage operates over a low-resolution representation space, it remains computationally tractable while facilitating global information exchange across views. Overall, this scalable design significantly reduces computational and memory overhead and allows for the incorporation of more viewpoints, thereby improving reconstruction fidelity.

We have comprehensively evaluated the proposed method on the large-scale datasets, RealEstate10K [61] and DL3DV [33]. The experimental results demonstrate that *iLRM* achieves superior rendering quality while significantly reducing computational and memory overhead compared to recently proposed feed-forward Gaussian models. For instance, on the RealEstate10K dataset, *iLRM* improved the PSNR by ~ 3 dB over the state-of-the-art methods, such as GS-LRM [59] and DepthSplat [53], by leveraging more views (8 views vs. 2 views) with only less than half of the computation time (0.028s vs. 0.065s). We further demonstrated that our method can generalize well across diverse scenes and viewpoint configurations, making it suitable for practical, large-scale 3D reconstruction tasks.

2 Related Works

2.1 Feed-forward 3D Gaussian Splatting

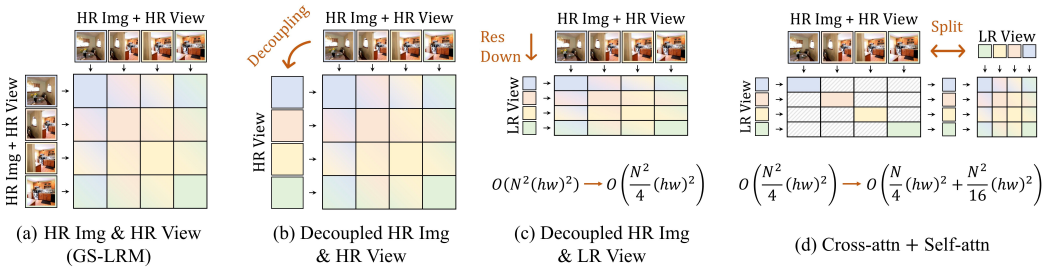
Feed-forward 3D Gaussian Splatting [5, 11, 53, 39, 59, 47, 48, 54] capitalizes on robust priors learned from extensive datasets to estimate Gaussian primitive parameters and synthesize novel view images using sparse input data. PixelSplat [5] and LatentSplat [51] predict Gaussians from image features using an epipolar line sampling method to enhance geometric accuracy, while MVSplat [11] and MVSGaussian [35] construct cost volumes through a plane-sweep stereo approach. In a further development, Flash3D [46] and DepthSplat [53] introduce a pre-trained depth estimation model [42, 55], which improves the robustness of the spatial positions of 3D Gaussians. In contrast, GS-LRM [59] and Long-LRM [62] minimize reliance on explicit 3D priors by leveraging large-scale data-driven priors and pre-trained 2D foundation models [3, 40, 55].

While demonstrating strong results, a major limitation of all the aforementioned approaches lies in their non-scalable architectural design, which restricts their ability to effectively leverage a large number of input views. Moreover, the one-shot generation strategy, which produces 3D representations in a single forward pass, often fails to capture complex geometric details and fine 3D consistency, making them suboptimal for high-quality 3D reconstruction. We address these limitations by proposing an iterative 3D reconstruction framework and scalable architectural designs.

Iterative refinements. Our work is also closely related to recent methods that adopt iterative refinement strategies, such as G3R [13] and Gen-Den [39]. Both utilize actual gradients to update their representations more precisely. While promising, these approaches require additional computational burden for rendering multiple images per training iteration, and relying solely on gradients may risk overlooking valuable information present in the raw input images. Nonetheless, exploring how to incorporate gradient information remains an interesting direction for future work.

2.2 3D Representations from Embeddings

Inspired by previous generative approaches [21, 28, 4], recent works [25, 6, 20] have investigated the synthesis of 3D representations directly from learnable embeddings, guided by input image supervision. This paradigm leverages the expressive capacity of latent spaces to encode rich geometric priors, which act as structural templates that guide the reconstruction process. Such approaches



(a) HR Img & HR View (GS-LRM) (b) Decoupled HR Img & HR View (c) Decoupled HR Img & LR View (d) Cross-attn + Self-attn

Figure 2: The proposed scalable architectural designs by decoupling viewpoint and image tokens, and modeling the global interactions via cross- and self-attentions (N: # views, $h = H/p, w = W/p$).

offer notable flexibility, allowing rendering from arbitrary viewpoints and adaptation to varying space scales and camera poses. However, both LRM [25] and Lara [6] are limited to object-centric representations, restricting scalability to complex scenes involving multiple objects or large spatial layouts. The recently proposed Quark [20] also utilizes learnable embeddings to fuse visual cues from multiple images, demonstrating compelling results, but its representation is confined to the target view [52, 35, 26], lacking an explicit and persistent 3D reconstruction.

In contrast to previous works, we construct scene-level explicit 3D representations from viewpoint embeddings by decoupling the generation of Gaussians from the input images. This separation enables iterative refinement of the embeddings using low-level visual features and provides flexible control over the density of the 3D representation, independent of the input image resolution.

3 Method

In this section, we provide a comprehensive explanation of our method, *iLRM*. We begin with the motivation and problem statement in Sec. 3.1, followed by a detailed presentation of the architectural design in Sec. 3.2, and finally training objectives in Sec. 3.3.

3.1 Motivation and Problem Statement

Existing generalizable 3D Gaussian reconstruction methods process multi-view images in an end-to-end fashion, often employing techniques such as epipolar line sampling [5], plane-sweep stereo [11, 12, 53], or full-resolution attention [59, 54, 62] to enforce multi-view consistency. While effective, these strategies introduce significant computational and memory overhead, limiting their scalability.

To address these challenges, we propose *iLRM*, a novel feed-forward 3D reconstruction framework that decouples Gaussian generation from direct dependence on input images. Instead of generating pixel-aligned Gaussians, *iLRM* initializes viewpoint-centric embeddings as the basis for constructing the 3D scene. These embeddings are then iteratively refined via cross-attention with multi-view image features, enabling the model to efficiently fuse geometric and appearance cues across views.

We start with N multi-view images $\{I_i\}_{i=1}^N$ and their corresponding camera poses $\{C_i\}_{i=1}^N$. Based on this setup, our goal is to train a model f_θ that maps a set of camera viewpoints to 3D Gaussians, leveraging the associated multi-view images as visual cues throughout the reconstruction pipeline. More formally,

$$f_\theta : \{(C_i, I_i)\}_{i=1}^N \mapsto \{(\mu_k, \alpha_k, \Sigma_k, c_k)\}_{k=1}^{H^v W^v N}, \tag{1}$$

where f_θ is modeled as a feed-forward network with the model parameter θ. $\mu_k, \alpha_k, \Sigma_k, c_k$ are attributes of 3D Gaussians, representing the mean, opacity, covariance, and color, respectively, while H^v and W^v denote the height and width of the generated Gaussians for each camera viewpoint. It is important to note that they do not correspond to the resolution of the input images. Following prior works, we train our model using held-out target images along with their corresponding camera poses, enabling high-quality novel view synthesis.

3.2 Architectural Design

We propose an end-to-end transformer that directly regresses 3D Gaussian parameters from viewpoint embeddings. To compensate for the absence of direct image input, we enrich these embeddings at each layer via cross-attention with multi-view image features. The resulting embeddings are further refined through self-attention to capture global dependencies across viewpoints.

Viewpoint tokenization. Following previous works [48, 59, 26], we employ a Plücker ray embedding for each input view using the camera poses. Specifically, given the intrinsic, rotation, and translation, we construct the Plücker ray embeddings for each viewpoint. We then divide these viewpoint embeddings into non-overlapping patches of size $p \times p$, and reshape each patch into a 1D vector, resulting in a tensor of shape $H^v W^v / p^2 \times 6p^2$. Then, we encode it using a single linear layer to produce viewpoint tokens, $V_i^{(0)} \in \mathbb{R}^{H^v W^v / p^2 \times d}$. Plücker coordinates inherently capture spatial variations across pixels and views, allowing them to effectively differentiate between patches. As a result, we do not utilize additional positional embeddings.

Multi-view image tokenization. For each input view image, which provides visual guidance to the reconstruction process, we extract both image features and corresponding pose information. Specifically, we divide an input image into non-overlapping patches and obtain two sets: RGB image patches and Plücker ray patches. These are then concatenated and linearly projected to construct the image patch tokens, $S_i \in \mathbb{R}^{HW/p^2 \times d}$,

$$S_{ij} = \text{Linear}(\text{concat}(I_{ij}, P_{ij})) \in \mathbb{R}^d, \tag{2}$$

where $I_{ij} \in \mathbb{R}^{3p^2}$, $P_{ij} \in \mathbb{R}^{6p^2}$ represent the flattened j-th RGB image and Plücker ray patches for the i-th view, respectively, and HW/p^2 is the number of tokens for each input view image.

Scalable multi-view context modeling. Fig. 2-(a) shows the typical feed-forward 3D methods [59, 11, 53] using transformer architecture, which perform full attention across multi-view images, incurring a quadratic increase in computational cost with respect to both the number of views and the image resolution. Fig. 2-(b) depicts our decoupling approach. Thanks to the decoupling technique, we can reduce the viewpoint resolution while still leveraging high-resolution multi-view images (Fig. 2-(c)). We further decrease the computation cost by two-stage multi-view context modeling, per-view cross-attention and viewpoint self-attention (Fig. 2-(d)). For example, given 16 input images with a resolution of 256 $\times$ 256 and a patch size of 8, the relative computational cost follows the ratio (1:1:0.25:0.08, Fig. 2-(a):(b):(c):(d)), highlighting that our proposed method can accommodate more input views with significantly less computational burden.

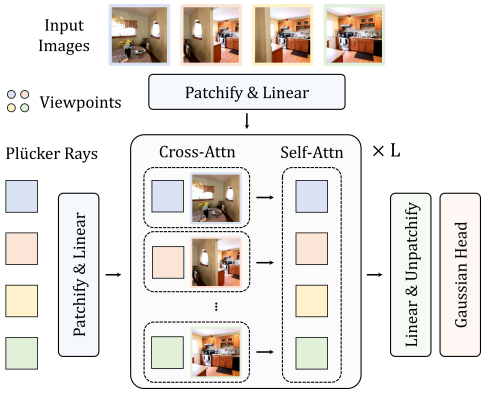


Figure 3: Network architecture.

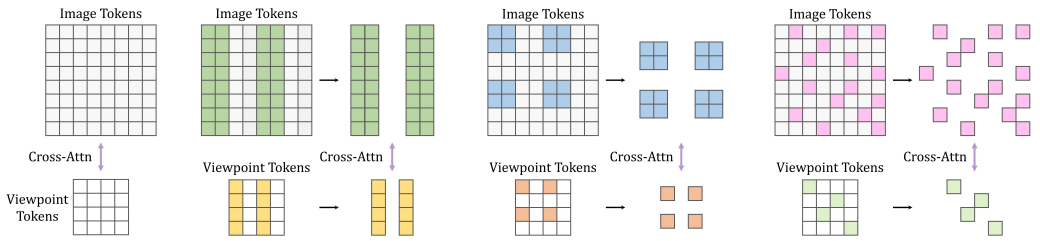
Update block. Given a set of viewpoint tokens, we formulate the problem as an iterative refinement process, where the viewpoint tokens are progressively updated through interactions with multi-view image tokens, ultimately leading to more accurate and spatially consistent 3D Gaussian Splatting. As shown in Fig. 3, our model consists of multiple transformer modules, each comprising one cross-attention layer followed by one self-attention layer.

$$\tilde{V}_i^{(l-1)} = \text{cross-attn}^{(l)}(V_i^{(l-1)}, S_i), \tag{3}$$

$$\{V_i^{(l)}\}_{i=1}^N = \text{self-attn}^{(l)}(\{\tilde{V}_i^{(l-1)}\}_{i=1}^N), \tag{4}$$

where the superscript (l) denotes the layer index. In the cross-attention layers, the viewpoint tokens are refined by the visual information from their corresponding image tokens. In the self-attention layers, the viewpoint tokens interact with each other to refine and enhance their representations by incorporating global contextual information. Note that we use separate model parameters for the update blocks at different layers.

Token uplifting. Standard cross-attention operations are typically applied between token sets with similar spatial resolutions, which allows for a straightforward one-to-one correspondence between queries and keys. In our setting, however, the cross-attention between viewpoint and image tokens involves mismatched resolutions: we intentionally use lower-resolution (LR) viewpoint tokens compared to image tokens to improve architectural scalability and enhance the efficiency of 3D Gaussian generation.



(a) Full Cross-Attn (b) Half Cross-Attn (c) Quarter Cross-Attn (d) Random Quater Cross-Attn

Figure 4: Various mini-batch cross-attention schemes.

This reduction in viewpoint token resolution, while beneficial for scalability and efficiency, may constrain their ability to fully capture and integrate the rich information present in the high-resolution (HR) image tokens. To bridge this gap, we propose a token uplifting strategy that aligns the LR viewpoint tokens with the HR image tokens. Specifically, each LR viewpoint token is lifted by a linear query layer that increases its feature dimension by a factor of k, resulting in the viewpoint token tensor of shape $H^v W^v/p^2 \times dk$. Then, it is reshaped to expand the token to the shape of $H^v W^v k/p^2 \times d$, where each original token now corresponds to k finer-grained query tokens, allowing for better capture of visual correspondence during attention against image tokens.

After performing cross-attention with HR image tokens serving as keys and values, the updated tokens tensor of shape $H^v W^v k/p^2 \times d$ are reshaped back to the original viewpoint tensor of shape $H^v W^v/p^2 \times dk$, and subsequently projected back to the original embedding dimension via a linear projection layer, resulting in the refined viewpoint token tensor of shape $H^v W^v/p^2 \times d$. This technique preserves the updated information while maintaining computational efficiency in subsequent self-attention layers. Balancing representational capacity and efficiency, we empirically set the expansion factor to $k = 2$.

Mini-batch cross-attention. In our architecture, viewpoint tokens are iteratively updated at each layer based on information from the image tokens via cross-attention. The proposed decoupled viewpoint token design allows us to arbitrarily reduce the number of viewpoint tokens for improved scalability, whereas the resolution of image tokens remains fixed due to their spatial nature. Consequently, the primary computational bottleneck in cross-attention lies in the high-resolution image tokens.

To address this, we propose several efficient cross-attention schemes, as illustrated in Fig. 4, aimed at improving scalability without sacrificing performance. Our design is conceptually inspired by mini-batch gradient descent in optimization, where only a subset of data points is sampled in each iteration to reduce computational cost. Similarly, our mechanism selectively samples subsets of both image tokens and viewpoint tokens during cross-attention. While random token sampling (Fig. 4-(d)) is ideal in theory, it complicates efficient implementation. To mitigate this, we design structured sampling strategies that are simple to implement and demonstrate strong empirical performance.

Decoding into 3D Gaussians. After the final self-attention layer, we decode the final layer's viewpoint tokens, $V_i^{(L)}$, into Gaussian parameters through a single linear layer and apply post-activation functions. For a detailed description, please refer to our supplementary materials.

3.3 Training Objectives

After generating 3D Gaussians from viewpoint tokens, we rasterize them to the target viewpoint to obtain rendered images, $\hat{I}_t$. These rendered images are then supervised using ground-truth images, I_t, through MSE loss and perceptual loss [7, 32] using a pretrained VGG network [45]. For each training scene, our training loss function is as follows.

$$\mathcal{L}_{\text{total}} = \sum_{t \in \mathcal{T}} \mathcal{L}_{\text{MSE}}(\hat{I}_t, I_t) + \lambda \mathcal{L}_{\text{perceptual}}(\hat{I}_t, I_t), \tag{5}$$

where $\mathcal{T}$ is a set of target view indices, λ is a weighting factor that balances the contribution of the perceptual loss relative to the MSE loss, which we set to 0.5.

Figure 5: Qualitative comparison of novel view synthesis on the RE10K dataset.

Figure 6: Qualitative comparison of novel view synthesis on the DL3DV dataset.

4 Experiments

4.1 Datasets

We train our model on two large-scale datasets: RealEstate10K (RE10K) [61], DL3DV [33], and evaluate it on three datasets, including ACID [34], which is used only for evaluation. For RE10k, we use 67,477 training and 7,289 testing scenes; for DL3DV, 9,568 training and 140 testing scenes, consistent with previous works [5, 11, 53]. Lastly, for ACID, we use only the test dataset, amounting to 1,972 scenes. These datasets cover a diverse range of scenes, including indoor and outdoor real estate videos (RE10K), aerial views of natural landscapes (ACID), and various real-world video scenes (DL3DV). We use an image resolution of 256×256 for the RE10K and ACID datasets, while for the DL3DV dataset, we use a resolution of 256×448 and 512×960.

4.2 Implementation and Training Details

Our model consists of 12 update layers, each containing one cross-attention block and one self-attention block. Inside each attention module, we adopt a pre-normalization method with LayerNorm [1] and QK-Norm [23] method with an RMSNorm [58] layer. Also, each of them utilizes multi-head attention [49] with 12 heads and two linear layers with GELU [22] activation. We set the hidden dimension of every linear layer to $d = 768$, and use a patch size of $p = 8$.

To enhance training efficiency in terms of both memory consumption and computational speed, we integrate several optimization techniques. Specifically, we utilize FlashAttention-2 [14] to improve attention computation efficiency, apply gradient checkpointing [8] to reduce memory overhead, and adopt mixed-precision training with BFloat16 to accelerate computations while maintaining numerical stability. During training, we sample a wide range of input viewpoints based on the number of frames, and when necessary, apply farthest point sampling (FPS) [43] using their camera positions. For each training example across all datasets, we sample 6 target views. For training on the DL3DV dataset, we start from the pretrained model on the RE10K dataset, following the approach in [53]. For more details, please refer to the supplementary material.

4.3 Evaluation

We compare our model with recent generalizable 3D reconstruction methods [5, 11, 59, 53, 39]. For evaluation, we follow the evaluation settings from [5, 56] for RE10K and [53] for DL3DV. Additionally, when utilizing more viewpoints than the evaluation protocol, we sample extra viewpoints between the input viewpoints, ensuring that these samples do not overlap with the target indices. We denote our various viewpoint settings as $(V, H/F, F)$, where V is the number of viewpoints, and the following entries indicate the resolutions of viewpoint and image tokens (H: half-resolution, F:

full-resolution). For example, a setting of $(2, H, F)$ indicates two viewpoints with half-resolution viewpoint tokens and full-resolution image tokens. MC refers to our quarter mini-batch cross-attention (Fig. 4-(c)). Note that our 2-view full-resolution setting $(2, F, F)$ does not include token uplifting, as the viewpoint and image resolutions are identical.

4.4 Quantitative Results

We report quantitative results for the RE10K dataset in Tab. 1, for the DL3DV dataset in Tab. 2, and cross-dataset generalization results in Tab. 3. Our method consistently outperforms the baselines across various viewpoint configurations, including inference speed and memory usage, while achieving efficient scene representation with fewer Gaussians. These improvements are also reflected in the qualitative results shown in Fig. 5 and Fig. 6, where our method produces sharper view synthesis compared to the baselines, which exhibit noticeable artifacts. Note that, for the RE10K dataset, we improved the PSNR by ~ 3 dB over the state-of-the-art methods by leveraging more views (8 views vs. 2 views) with only less than half of the computation time (0.028s vs. 0.065s). For the DL3DV dataset, we improved the PSNR by ~ 4 dB over the state-of-the-art methods under the comparable computational budget, efficiently leveraging four times more views (6 vs. 24 views).

Method	#Param (M)	PSNR ↑	SSIM ↑	LPIPS ↓	# of Gaussians	Time (S)
pixelSplat [5]	125	25.89	0.858	0.142	131,072	0.101
MVSplat [11]	12	26.39	0.869	0.128	131,072	0.047
GS-LRM* [59]	300	28.10	0.892	0.114	131,072	-
DepthSplat [53]	354	27.47	0.889	0.114	131,072	0.065
Gen-Den [39]	28	27.08	0.879	0.120	347,072	0.224
Ours $(2, F, F)$	171	28.17	0.894	0.115	131,072	**0.025**
Ours $(4, H, F)$	185	30.01	0.918	0.098	**65,536**	0.027
Ours-MC $(4, H, F)$	185	29.75	0.915	0.101	**65,536**	0.027
Ours $(8, H, F)$	185	**31.01**	**0.930**	**0.088**	131,072	0.028
Ours-MC $(8, H, F)$	185	30.69	0.927	0.090	131,072	0.030

Table 1: Quantitative comparisons on the RE10K [61] dataset with various viewpoint configurations. Inference time is the reconstruction time (not rendering speed), using a single NVIDIA RTX 4090 GPU. * indicates methods for which the official code and the pretrained models are not available.

Method	Views	PSNR ↑	SSIM ↑	LPIPS ↓	# of Gaussians	Time (S)	Memory (GB)
MVSplat [11]	6	22.93	0.775	0.193	688,128	0.279	5.87
DepthSplat [53]		24.19	0.823	0.147	688,128	0.102	3.55
Ours	$(6, H, F)$	25.07	0.812	0.189	**172,032**	**0.031**	**1.40**
	$(11, H, F)$	26.47	0.852	0.155	315,392	0.051	1.59
	$(24, H, F)$	**27.97**	**0.886**	**0.128**	688,128	0.123	2.01

Table 2: Quantitative comparisons on the DL3DV [33] dataset with various view configurations. Inference time is the reconstruction time (a single NVIDIA RTX 4090), and memory consumption is measured during the reconstruction (a single forward pass).

4.5 High-resolution Experiment

We present high-resolution experimental results (512×960) in Table 4 and Figure 7. Our method is compared with DepthSplat [53] using a 100-frame interval and 12 input images/viewpoints. The relatively small gap in inference time is that DepthSplat had to use a smaller model, as the computational cost for processing high-resolution images was significantly higher.

4.6 Computational Costs of Training

We report detailed comparisons of computational costs during training in Tab. 5. The iteration time is measured under the same setting: half-resolution 8 viewpoints $(8, H, F)$, and a batch size of 16 on a single RTX 4090 GPU. For memory comparison, to provide a clearer analysis, all models are run without gradient checkpointing on a single H100 GPU. Lastly, we present a theoretical

Method	ACID [34]			DL3DV [33]		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
MVSplat [11]	28.15	0.841	0.147	22.65	0.737	0.191
DepthSplat [53]	28.37	0.847	0.141	24.28	0.813	0.147
Gen-Den [39]	28.61	0.847	0.141	22.92	0.750	0.188
Ours $(2, F, F)$	28.73	0.847	0.153	25.02	0.815	0.154
Ours $(4, H, F)$	29.80	0.871	0.140	27.60	0.870	0.127
Ours-MC $(4, H, F)$	29.60	0.867	0.144	27.38	0.865	0.132
Ours $(8, H, F)$	**30.51**	**0.886**	**0.131**	**28.98**	**0.897**	**0.112**
Ours-MC $(8, H, F)$	30.24	0.883	0.134	28.78	0.895	0.114

Table 3: Cross-dataset generalization results on the ACID [34] and DL3DV [33] datasets. All models are trained on the RE10K [61] training set and evaluated on each dataset without fine-tuning.

Method	Views	PSNR ↑	SSIM ↑	LPIPS ↓	# of Gaussians	Time (S)	Memory (GB)
DepthSplat [53]	12	21.38	0.739	**0.265**	5,898,240	0.504	24.93
Ours	$(12, H, F)$	**23.92**	**0.764**	0.274	**1,474,560**	**0.415**	**3.52**

Table 4: Quantitative comparisons on the DL3DV [33] dataset on high-resolution setting. Inference time and memory consumption are measured only during the Gaussian generation stage, excluding the rendering process on a single H100 GPU.

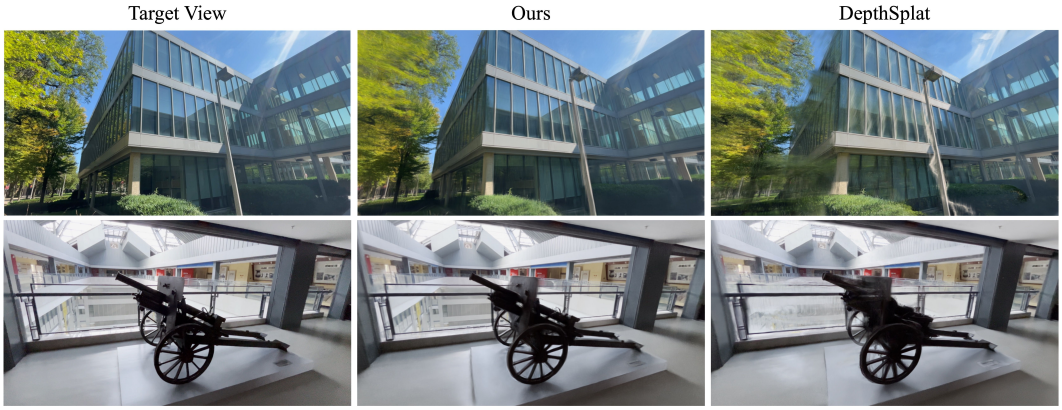


Figure 7: Qualitative comparison of novel view synthesis on the high-resolution (512×960) DL3DV dataset.

comparison of FLOPs that further underscores the efficiency of our method, with only a marginal drop in performance. For detailed calculation, please refer to our supplementary material.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	Iteration time (S)	Training Memory (GB)	GFLOPs
Baseline	**30.15**	**0.920**	**0.097**	1.51	62.5	3.83
w/ Half Cross-attn	30.02	0.919	0.098	1.13	47.4	1.71
w/ Quater Cross-attn	29.85	0.916	0.101	**0.94**	**39.0**	**0.81**

Table 5: Quantitative comparisons of our mini-batch cross-attention on the RE10K [61] dataset.

4.7 Ablations and Analysis

Tab. 6 presents the ablations on the number of layers. All variants are trained under half-resolution 4 viewpoints setting $(4, H, F)$, with a batch size of 16 on a single RTX 4090 GPU. The results demonstrate the scalability of our transformer-based architecture, showing consistent performance gains as the number of layers increases. Also, from the perspective of the iterative refinement procedure, increasing the number of layers can be interpreted as introducing more optimization steps, which aligns with our intuition that deeper refinement leads to more accurate 3D representations.

	# Params	PSNR ↑	SSIM ↑	LPIPS ↓
12 layers (base)	185M	**28.96**	**0.902**	**0.111**
9 layers	139M	28.74	0.898	0.114
6 layers	94M	28.42	0.893	0.119
3 layers	48M	27.83	0.882	0.130

Table 6: Ablations on model size.

	PSNR ↑	SSIM ↑	LPIPS ↓
Baseline	**28.96**	**0.902**	**0.111**
w/o token uplifting	28.69	0.898	0.116
w/o self-attention	23.29	0.753	0.222
w/ group-attention	28.76	0.899	0.115

Table 7: Ablations on model architecture.

In Tab. 7, we present the ablation results on key architectural components to validate the effectiveness of each design choice. All experiments are conducted under the same configuration and settings as the model size ablation with 24 layers baseline.

1) Token uplifting. Removing the token uplifting mechanism leads to a drop in performance across all metrics compared to baseline. This validates the importance of expanding low-resolution view tokens before cross-attention with high-resolution image tokens. Without this step, the model struggles to capture fine-grained spatial correspondences, resulting a degraded reconstruction quality.

2) Self-attention. To ensure a fair comparison, we replaced all self-attention layers with cross-attention layers rather than simply removing them, maintaining a comparable parameter count. The performance dropped significantly, highlighting the essential role of self-attention in capturing global dependencies and enhancing multi-view awareness among viewpoint embeddings. Without self-attention, the model struggles to integrate contextual information across different viewpoints, resulting in poor convergence and reconstruction quality.

3) Group-attention This variant replaces the per-viewpoint cross-attention mechanism with a group-attention approach, where all viewpoint tokens and image tokens are concatenated and jointly processed through a cross-attention block (Fig. 2-(c)). Unlike our default design, group-attention introduces global interactions across all views. While this mechanism can increase the expressive capacity between multiple viewpoints, it incurs quadratic complexity with respect to the number of views. However, the increased computational cost does not yield performance gains, suggesting that separating the roles—using cross-attention for localized image-view interactions and self-attention for global refinement across viewpoints—leads to a more efficient and effective architecture.

5 Limitations

One limitation of this work is the computational bottleneck associated with self-attention across multiple input views. Although we significantly reduce the computational cost by utilizing compact viewpoint embeddings, challenges may arise as the number of input views increases considerably. In this study, aiming for scalable feed-forward 3D models, we present the first implementation of the proposed framework, a feed-forward 3D model that iteratively refines 3D representations by leveraging high-resolution image information at every layer. Further development of more scalable alternatives to self-attention, such as hierarchical and sparse attention mechanisms or other efficient processing strategies, would be a valuable direction for future research.

Additionally, our model requires known camera poses, typically obtained from structure-from-motion (SfM) [44, 41] methods such as COLMAP or from precisely synthesized datasets. However, these poses can be error-prone and are often difficult to acquire at scale. To enhance the practicality of our model and enable the use of giant-scale raw video datasets [9], extending it to operate in a pose-free setting [56, 24, 27] remains an important and open direction for future work.

6 Conclusion

In this work, we present an iterative Large 3D Reconstruction Model (*iLRM*), a feed-forward architecture that reflects per-scene optimization-based schemes, by stacking multiple update layers composed of cross- and self-attention modules. By decoupling Gaussian representations from input images and splitting the update mechanism into per-view interactions with image features and global context aggregation over compact viewpoint embeddings, *iLRM* enables efficient, scalable, and high-quality 3D reconstruction across diverse scenes. We believe that *iLRM* lays a strong foundation for future research in feed-forward 3D reconstruction.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[4] 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. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16123–16133, 2022.

[5] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[6] Anpei Chen, Haofei Xu, Stefano Esposito, Siyu Tang, and Andreas Geiger. Lara: Efficient large-baseline radiance fields. In *European Conference on Computer Vision*, pages 338–355. Springer, 2024.

[7] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1511–1520, 2017.

[8] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[9] Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, et al. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13320–13331, 2024.

[10] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, pages 422–438. Springer, 2024.

[11] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, 2025.

[12] 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. *arXiv preprint arXiv:2411.04924*, 2024.

[13] Yun Chen, Jingkang Wang, Ze Yang, Sivabalan Manivasagam, and Raquel Urtasun. G3r: Gradient guided generalizable reconstruction. In *European Conference on Computer Vision*, pages 305–323. Springer, 2024.

[14] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. *arXiv preprint arXiv:2307.08691*, 2023.

[15] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 2024.

[16] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[17] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. Roma: Robust dense feature matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19790–19800, 2024.

[18] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37:140138–140158, 2024.

[19] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision*, 2024.

[20] John Flynn, Michael Broxton, Lukas Murmann, Lucy Chai, Matthew DuVall, Clément Godard, Kathryn Heal, Srinivas Kaza, Stephen Lombardi, Xuan Luo, et al. Quark: Real-time, high-resolution, and general neural view synthesis. *ACM Transactions on Graphics*, 2024.

[21] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[22] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[23] Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers. *arXiv preprint arXiv:2010.04245*, 2020.

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

[25] 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.

[26] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. Lvsm: A large view synthesis model with minimal 3d inductive bias. *arXiv preprint arXiv:2410.17242*, 2024.

[27] 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. *arXiv preprint arXiv:2411.17190*, 2024.

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

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

[30] Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Optimized minimal 3d gaussian splatting. *arXiv preprint arXiv:2503.16924*, 2025.

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

[32] Zhengqi Li, Wenqi Xian, Abe Davis, and Noah Snavely. Crowdsampling the plenoptic function. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 178–196. Springer, 2020.

[33] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[34] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[35] Tianqi Liu, Guangcong Wang, Shoukang Hu, Liao Shen, Xinyi Ye, Yuhang Zang, Zhiguo Cao, Wei Li, and Ziwei Liu. Mvsgaussian: Fast generalizable gaussian splatting reconstruction from multi-view stereo. In *European Conference on Computer Vision*, pages 37–53. Springer, 2024.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[37] Tao Lu, Ankit Dhiman, R Srinath, Emre Arslan, Angela Xing, Yuanbo Xiangli, R Venkatesh Babu, and Srinath Sridhar. Turbo-gs: Accelerating 3d gaussian fitting for high-quality radiance fields. *arXiv preprint arXiv:2412.13547*, 2024.

[38] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, 2024.

[39] Seungtae Nam, Xiangyu Sun, Gyeongjin Kang, Younggeun Lee, Seungjun Oh, and Eunbyung Park. Generative densification: Learning to densify gaussians for high-fidelity generalizable 3d reconstruction. *arXiv preprint arXiv:2412.06234*, 2024.

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

[41] Linfei Pan, Dániel Baráth, Marc Pollefeys, and Johannes L Schönberger. Global structure-from-motion revisited. In *European Conference on Computer Vision*, pages 58–77. Springer, 2024.

[42] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10106–10116, 2024.

[43] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 2017.

[44] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[46] Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, Joao F Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image. *arXiv preprint arXiv:2406.04343*, 2024.

[47] Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[48] 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 *European Conference on Computer Vision*, 2025.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[50] Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. Freesplat: Generalizable 3d gaussian splatting towards free view synthesis of indoor scenes. *Advances in Neural Information Processing Systems*, 37, 2025.

[51] Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction. In *European Conference on Computer Vision*, pages 456–473. Springer, 2024.

[52] Haofei Xu, Anpei Chen, Yuedong Chen, Christos Sakaridis, Yulun Zhang, Marc Pollefeys, Andreas Geiger, and Fisher Yu. Murf: Multi-baseline radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[53] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. *arXiv preprint arXiv:2410.13862*, 2024.

[54] 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. *arXiv preprint arXiv:2403.14621*, 2024.

[55] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[56] 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.

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

[58] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 2019.

[59] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, 2025.

[60] Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19680–19690, 2024.

[61] 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.

[62] 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. *arXiv preprint arXiv:2410.12781*, 2024.

A Additional Implementation Details

We initialize model weights using a zero-mean normal distribution with a standard deviation of 0.02. Bias terms are omitted in all Linear and normalization layers. The model is trained using the AdamW [36] optimizer with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.095$. A weight decay of 0.05 is applied to all parameters except the weights of LayerNorm [1]. We use a cosine learning rate schedule with a peak learning rate of 2e-4 and a warmup of 2500 iterations. Our training setup largely follows the configuration proposed in [59, 26].

The 8-view half-resolution viewpoint setting *(8, H, F)* is trained on 8 H100 GPUs with a total batch size of 256 for 50,000 iterations. Similarly, the 4-view half-resolution viewpoint setting *(4, H, F)* is trained on 8 RTX 4090 GPUs with a total batch size of 128 for 100,000 iterations. The mini-batch cross-attention variants were also trained with the equivalent computational budgets for each viewpoint setting using the RealEstate 10K [61] dataset.

For training on the DL3DV dataset [33], we initialize from the pretrained *(8, H, F)* model trained on the RE10K dataset [61], and finetuned on 8 H100 GPUs with a total batch size of 96 for 100,000 itertions. During training, the number of input viewpoints is randomly sampled between 6 and 11 to expose the model to varying numbers of viewpoints/images [53]. An additional 50,000 iterations are performed for high-resolution finetuning with varying numbers of viewpoints/images.

Gaussian representations. After the final self-attention layer, the viewpoint features are decoded into Gaussian parameters using a single linear layer with an output dimension of 16. The Gaussian positions, denoted as μ, consist of 5 channels: 2 for the spatial *xy* offset and 3 for depth, *z*. The final depth is obtained by averaging the 3 depth channels. Opacity (α) is represented by a single channel. Covariance (Σ) is derived from 3 channels of scale and 4 channels of rotation. Finally, color (c) is represented using 3 channels. Higher-order spherical harmonics coefficients are not used in our method. The post-activation functions for each parameter follow the design of GS-LRM [59], except for the spatial *xy* offset, for which we constrain the range to lie within a single pixel of viewpoint resolution. We utilize gsplat [57], an open-source library for Gaussian Splatting [29] for a rasterizer.

Camera pose normalization. We normalize camera poses to align the scene into a consistent coordinate system and scale. First, we compute the average position and viewing directions (forward, down, and right) from the input camera extrinsics. These are used to build a new reference pose, which centers and aligns the scene. All camera extrinsics are then transformed into this reference frame. Finally, we scale the entire scene so that the largest camera distance is 1, ensuring the scene fits within a normalized space [59].

B Additional Architectural Details.

We provide the detailed figure of our token uplifting module in Fig. 8. Note that, to balance the model's representational capacity and computational efficiency, the length of the low-resolution viewpoint embeddings does not exceed that of the high-resolution image features.

Tokenization and normalization. After tokenizing the viewpoints and multi-view images using linear layers, both types of tokens are passed through a LayerNorm [1]. In each cross-attention layer, only the viewpoint tokens are further processed with a pre-normalization layer. Additionally, after the query and key linear projections, both tokens are passed through an extra normalization layer, referred to as the QK-Norm [23].

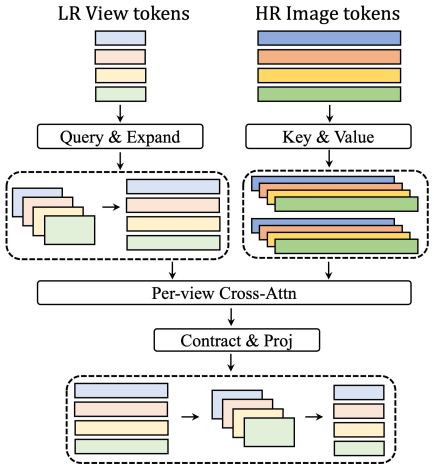


Figure 8: Token uplifting.

C Additional Evaluation Details.

When we utilize more input viewpoints (more than two in RealEstate10K [61] experiment compared to the baselines), we sample additional viewpoints/images evenly between the two endpoint indices, ensuring that these samples do not overlap with the target indices. For the DL3DV evaluation setting, we compare against the baselines using a 6-view configuration. In our 11-view configuration, we sample one additional viewpoint between every pair of consecutive baseline viewpoints. For 24 viewpoints setting, we finetuned the model with a fixed number of views for an additional 100,000 iterations.

DL3DV evaluation. For cross-dataset generalization on the DL3DV dataset, we use a baseline of 12 frames. For in-domain low-resolution evaluation, the baseline includes 50 frames in total.

D Computational Costs of Training

We provide a detailed theoretical calculation of the FLOPs for our mini-batch cross-attention mechanism. In this analysis, we limit the computation to a per-view, single cross-attention operation, excluding our token uplifting strategy (as it introduces a constant cost across all variations). Given a viewpoint token of shape (L_v, D) and an image token of shape (L_i, D), where L_v and L_i denote the token lengths and D is the hidden dimension, the FLOPs for the cross-attention operation are computed as:

$$4D^2(L_v + L_i) + 4L_vL_iD.$$

Assuming a hidden dimension of $D = 768$, an image resolution of 256×256, and a viewpoint resolution of 128×128, with a patch size of 8×8, the token lengths are computed as $L_i = 1024$ for the image tokens and $L_v = 256$ for the viewpoint tokens, based on the experimental configuration used in the RealEstate10K [61] dataset.

Thus, the computation becomes: baseline: **3.83 GFLOPs**; half cross-attention: **1.71 GFLOPs**; quarter cross-attention: **0.81 GFLOPs**.

E Additional Quantitative Results

Varying baseline range. We compare our model against recent generalizable 3D reconstruction methods [11, 53, 39] on the RealEstate10K [61] dataset, with a particular focus on handling varying degrees of camera overlap [56]. These overlap categories are determined using the dense feature matching method RoMA [17]. As shown in Tab. 8, our method, which efficiently handles a large number of input viewpoints/images, achieves superior performance compared to existing approaches, especially in challenging cases with small viewpoint overlap (i.e., wide-baseline settings).

	Small			Medium			Large			Average		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
MVSplat [11]	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 [53]	22.82	0.798	0.193	25.38	0.851	0.145	28.32	0.900	0.104	25.59	0.852	0.145
Gen-Den [39]	21.10	0.744	0.234	24.57	0.828	0.162	28.26	0.895	0.108	24.77	0.826	0.164
Ours $(2, F, F)$	23.66	0.809	0.189	26.20	0.858	0.145	28.82	0.901	0.110	26.33	0.858	0.146
Ours $(4, H, F)$	27.45	0.883	0.130	28.83	0.903	0.110	30.30	0.921	0.094	28.91	0.903	0.111
Ours-MC $(4, H, F)$	27.22	0.878	0.134	28.57	0.899	0.114	30.02	0.918	0.098	28.66	0.899	0.114
Ours $(8, H, F)$	**28.87**	**0.904**	**0.113**	**29.96**	**0.918**	**0.099**	**31.20**	**0.932**	**0.085**	**30.05**	**0.919**	**0.098**
Ours-MC $(8, H, F)$	28.72	0.903	0.114	29.73	0.916	0.101	30.85	0.929	0.088	29.80	0.916	0.101

Table 8: Quantitative comparisons on the RE10K [61] dataset under varying view overlap conditions.

Same number of Gaussians. We also validate the strength of our decoupling strategy in leveraging high-resolution images as visual cues while generating efficient and compact low-resolution 3D Gaussians. As discussed in our motivation, previous methods [5, 11, 59, 53, 39] require downsampling the input images to reduce the number of generated Gaussians, inherently coupling image resolution with representation density. To demonstrate the flexibility of our approach, we conduct an experiment in which all methods generate the same number of Gaussians using 4 viewpoints at half resolution. Specifically, the baseline methods [11, 53] follow a $(4, H, H)$ configuration, where both the number of viewpoints and image resolution are reduced. In contrast, our method adopts a $(4, H, F)$ setting, where we preserve high-resolution image inputs while generating low-resolution Gaussians, thanks

to our decoupled design. As shown in Tab. 9, our method surpasses the baselines in performance while requiring fewer computational resources in training, and faster inference speed, highlighting the practical advantages of our design. This result demonstrates the efficiency and the representational ability of our architecture, which effectively utilizes high-resolution visual cues, leading to superior reconstruction quality under the same output density without requiring expensive hardware. To train the baseline methods effectively with a large batch size (similar to ours), we run them on a single H100 GPU. Our method and MVSplat [11] are trained with a batch size of 16, while DepthSplat [53] is trained with a batch size of 12 due to memory constraints.

Method	Params (M)	Train GPU (#)	PSNR ↑	SSIM ↑	LPIPS ↓	# of Gaussians	Time (S)	Memory (GB)
MVSplat [11]	12	H100 (1)	27.53	0.889	0.116	65,536	0.048	**0.65**
DepthSplat [53]	354	H100 (1)	28.08	0.898	0.107	65,536	0.062	2.49
Ours	185	RTX 4090 (1)	28.96	0.902	0.111	65,536	**0.027**	1.22
Ours	185	RTX 4090 (2)	29.50	0.911	0.104	65,536	**0.027**	1.22
Ours	185	RTX 4090 (4)	**29.98**	**0.918**	**0.098**	65,536	**0.027**	1.22

Table 9: Quantitative comparisons under the same number of Gaussians on the RE10K [61] dataset. Inference time and memory consumption are measured only during the Gaussian generation stage, excluding the rendering process on a single RTX 4090 GPU.

Quarter resolution baselines. To evaluate different viewpoint configurations—specifically the resolution of each viewpoint—we additionally compare a quarter-resolution variant of the viewpoint inputs. All experiments in Tab. 10 are conducted using a single RTX 4090 GPU with a batch size of 16, 12 update layers, and trained for 100,000 iterations. While lowering the resolution of viewpoint inputs leads to a moderate drop in reconstruction quality, it significantly reduces the number of generated Gaussians, showing trade-off between accuracy and efficient representations.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	# of Gaussians
Ours $(8, H, F)$	**30.15**	**0.920**	**0.097**	131,072
Ours $(4, H, F)$	28.96	0.902	0.111	65,536
Ours $(8, Q, F)$	27.21	0.864	0.154	32,768
Ours $(4, Q, F)$	26.25	0.839	0.180	**16,384**

Table 10: Quantitative comparisons of different viewpoint configurations on the RE10K [61] dataset. Q denotes quarter resolution compared to the original image resolution.

Inference time and memory usage. We benchmark the inference time and peak GPU memory usage for generating Gaussians with varying numbers of input viewpoints/images. We use an image resolution of 256 $\times$ 448 and a viewpoint resolution of 128 $\times$ 224, following the DL3DV dataset experiment settings, and conduct measurements on a single RTX 4090 GPU. Note that, 50 target images are allocated as a bias in accordance with the evaluation setting presented in the main paper.

# of frames	2	4	6	8	12	16	32	64	128	256
Time (S)	0.027	0.028	0.031	0.039	0.054	0.081	0.181	0.458	1.285	4.065
Memory (GB)	1.29	1.36	1.40	1.45	1.62	1.76	2.31	3.42	5.65	10.11

Table 11: Inference time and memory usage with varying numbers of input viewpoints/images.

F Additional Qualitative Results

We present additional qualitative results in Fig. 9 and Fig. 10 corresponding to the RealEstate10K (RE10K) [61] and DL3DV [33] datasets, respectively. Also, Fig. 11 presents high-resolution visualizations from the DL3DV dataset.

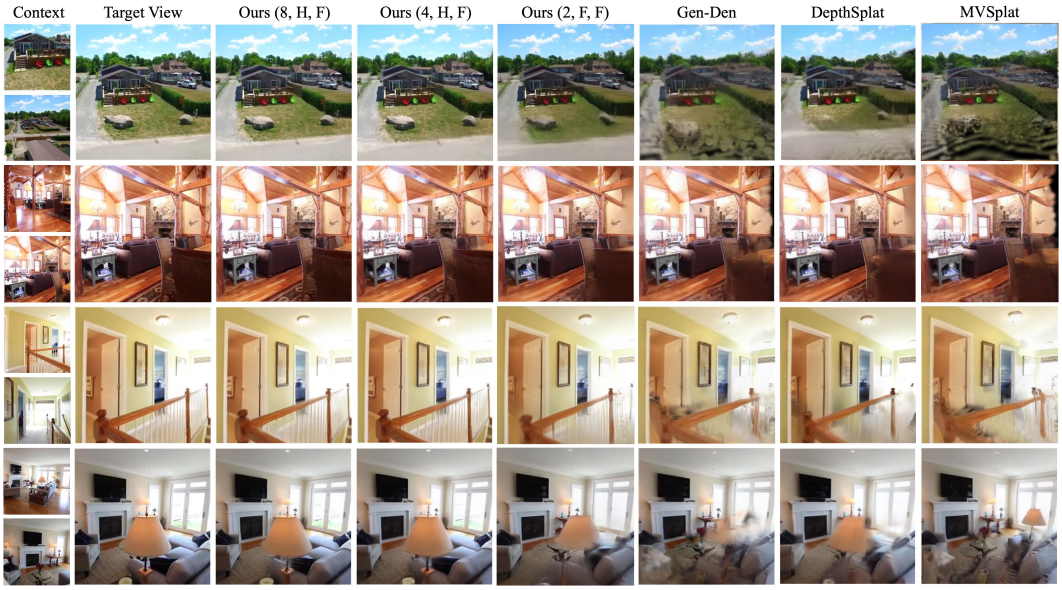


Figure 9: Qualitative comparison of novel view synthesis on the RE10K dataset.

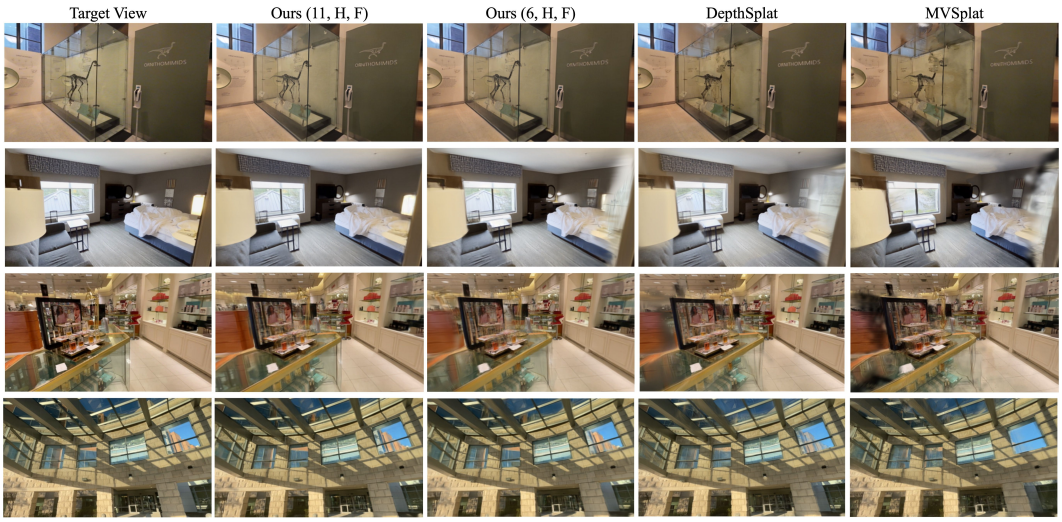


Figure 10: Qualitative comparison of novel view synthesis on the DL3DV dataset.

Figure 11: High-resolution novel view synthesis on the DL3DV dataset.