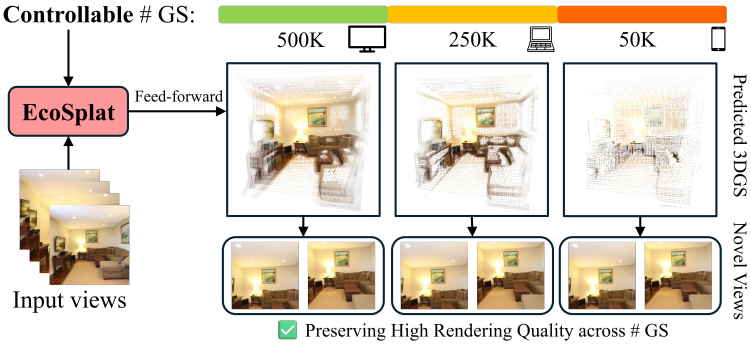
This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

EcoSplat: Efficiency-controllable Feed-forward 3D Gaussian Splatting from Multi-view Images

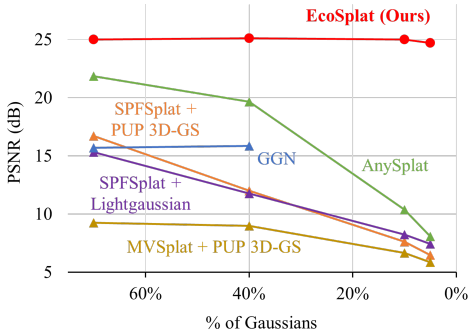
Jongmin Park[1*] Minh-Quan Viet Bui[1*]
Juan Luis Gonzalez Bello[2] Jaeho Moon[1] Jihyong Oh[3†] Munchurl Kim[1†]

[1]KAIST [2]Flawless AI [3]CMLab, Chung-Ang University

{jm.park, bvmquan, jaeho.moon, mkimee}@kaist.ac.kr
juanluis.gonzalez@flawlessai.com jihyongoh@cau.ac.kr
https://kaist-viclab.github.io/ecosplat-site/



(a) Concept diagram of our proposed EcoSplat



(b) NVS results across varying primitive counts

Figure 1. (a) EcoSplat is a feed-forward 3D Gaussian Splatting framework that enables explicit control over the number of output primitives. (b) It consistently outperforms state-of-the-art methods across a wide range of target primitive counts.

Abstract

Feed-forward 3D Gaussian Splatting (3DGS) enables efficient one-pass scene reconstruction, providing 3D representations for novel view synthesis without per-scene optimization. However, existing methods typically predict pixel-aligned primitives per-view, producing an excessive number of primitives in dense-view settings and offering no explicit control over the number of predicted Gaussians. To address this, we propose EcoSplat, the first efficiency-controllable feed-forward 3DGS framework that adaptively predicts the 3D representation for any given target primitive count at inference time. EcoSplat adopts a two-stage optimization process. The first stage is Pixel-aligned Gaussian Training (PGT) where our model learns initial primitive prediction. The second stage is Importance-aware Gaussian Finetuning (IGF) stage where our model learns rank primitives and adaptively adjust their parameters based on the target primitive count. Extensive experiments across multiple dense-view settings show that EcoSplat is robust and outperforms state-of-the-art methods under strict primitive-count constraints, making it well-suited for flexible downstream rendering tasks. Code and project page will be released.

1. Introduction

Novel View Synthesis (NVS) is a fundamental task in 3D vision, enabling applications in virtual reality (VR), augmented reality (AR), and cinematic production. Neural implicit representations [2, 3, 7, 16, 37, 38, 55] have shown impressive progress, yet their training and rendering remain computationally expensive. Recently, 3D Gaussian Splatting (3DGS) [28] has emerged as a state-of-the-art method, achieving both improved rendering quality and real-time rendering speeds. Its underlying framework has been further applied to dynamic reconstruction [31, 39, 48, 52], unconstrained image reconstruction [18], degraded in-

*Co-first authors (equal contribution).
†Co-corresponding authors.

puts [60], and geometric reconstruction [23]. Although 3DGS enables real-time rendering and fast reconstructions, it still requires substantial training time, often several minutes or even hours per scene for reconstruction, since it relies on per-scene optimization. To address this limitation, several feed-forward-based approaches [8, 11, 24, 27, 42, 50, 53, 57] have been proposed to directly predict the properties of 3D Gaussians (primitives) from multi-view images, leveraging large-scale datasets. In general, these approaches unproject the 2D pixels of each input image into a corresponding 3D primitive (i.e., *pixel-aligned* Gaussians) and aggregate the resulting primitives from all input views to represent the corresponding scene.

After the feed-forward reconstruction stage, the predicted 3D Gaussian primitives can be stored and reused for real-time rendering on end-user devices such as mobile phones or AR/VR headsets. In practical applications, reconstruction is typically performed once on the server-side devices, while the generated 3DGS must support multiple rendering requests across diverse end-user devices. However, existing feed-forward methods typically generate a large number of pixel-aligned primitives, especially for dense-view reconstructions, which can be impractical for resource-constrained devices. To mitigate this, several approaches [26, 59, 62] have attempted to reduce redundant Gaussians across multiple views through voxelization [26], graph-based aggregation [59], key-frame clustering [47], or opacity-based pruning [62]. Nonetheless, these methods *lack explicit control* over the target numbers of primitives *because they were not specifically designed for fine-level control over this number*, which is a crucial factor for flexible deployment in real-time and streamable applications. Their threshold-based pruning strategies (e.g., voxel size or opacity-based pruning) often yield scene-dependent and unstable primitive counts, resulting in a poor quality-efficiency trade-off. In practice, *explicit* control over the number of 3D Gaussian primitives is essential: without a mechanism to enforce a prescribed count, the final number of primitives varies across scenes and view counts, leading to unpredictable latency, memory, and bandwidth requirements. *Importantly*, it should be noted that our work is distinguished from the standardization activities of MPEG [25] and other academic works [12, 40] where 3D Gaussians are subject to *compression* via codecs while our approach is related to effective and controllable *compaction* of feed-forward 3DGS in terms of the number of the primitives.

To overcome these challenges, we propose EcoSplat, an Efficiency-controllable feed-forward 3D Gaussian Splatting framework for dense-view reconstruction. As shown in Fig. 1, EcoSplat maximizes rendering quality while explicitly controlling the number of 3D Gaussians to match a target number, all in a feed-forward manner. To achieve this, we introduce a two-stage training process consisting of Pixel-aligned Gaussian Training (PGT) and Importance-aware Gaussian Finetuning (IGF). In the first stage (PGT), we train our model to reliably predict pixel-aligned 3D Gaussians. In the second stage (IGF), we further train our model to adaptively predict the 3D Gaussian parameters conditioned on the target number of primitives. Specifically, in the IGF stage, we introduce an importance-aware opacity loss $\mathcal{L}_{\text{io}}$ to encourage the suppression of less informative primitives. Moreover, for improved training stability, we adopt a Progressive Learning on Gaussian Compaction (PLGC) strategy that gradually expands the sampling range of the target number of primitives during training. As a result of this training process, our EcoSplat can adaptively select the most important Gaussians for any target number of primitives during inference (Fig. 1-(b)). Our main contributions are summarized as follows:

- We introduce EcoSplat, the *first* feed-forward 3DGS framework that reconstructs a high-quality and robust set of Gaussian primitives while satisfying *explicit and controllable constraints on the target number of primitives*.
- We design a two-stage training process, including Pixel-aligned Gaussian Training (PGT) followed by Importance-aware Gaussian Finetuning (IGF), that enables the model to learn an *importance-aware primitive ranking* and achieve *controllable efficiency*.
- We demonstrate state-of-the-art efficiency-controllable NVS performance on dense-view benchmarks such as RealEstate10K [61] and ACID [34]. By achieving an optimal trade-off between primitive count and rendering quality, EcoSplat significantly outperforms previous feed-forward methods that lack fine-grained control over the number of primitives.

2. Related Work

Optimization-based NVS. Since the introduction of Neural Radiance Fields (NeRF) [37], the research on novel view synthesis (NVS) has made remarkable progress, with numerous extensions [4, 5, 7, 10, 16, 17, 19, 22, 38, 55]. 3D Gaussian Splatting (3DGS) [28] has emerged as an explicit scene representation approach that models a scene using millions of anisotropic Gaussian primitives and employs differentiable rasterization to achieve photorealistic novel views with real-time inference. Despite these advantages, 3DGS relies on per-scene optimization of Gaussians and cannot directly render novel views from input images without retraining. Very recently, several post-optimization techniques have been proposed to control the number of Gaussian primitives [15, 20]. PUP 3D-GS [20] computes a spatial sensitivity score to prune pretrained 3DGS models. LightGaussian [15] utilizes significance-based pruning, SH coefficient distillation, and vector quantization to reduce the 3DGS model size. However, similar to 3DGS [28], these

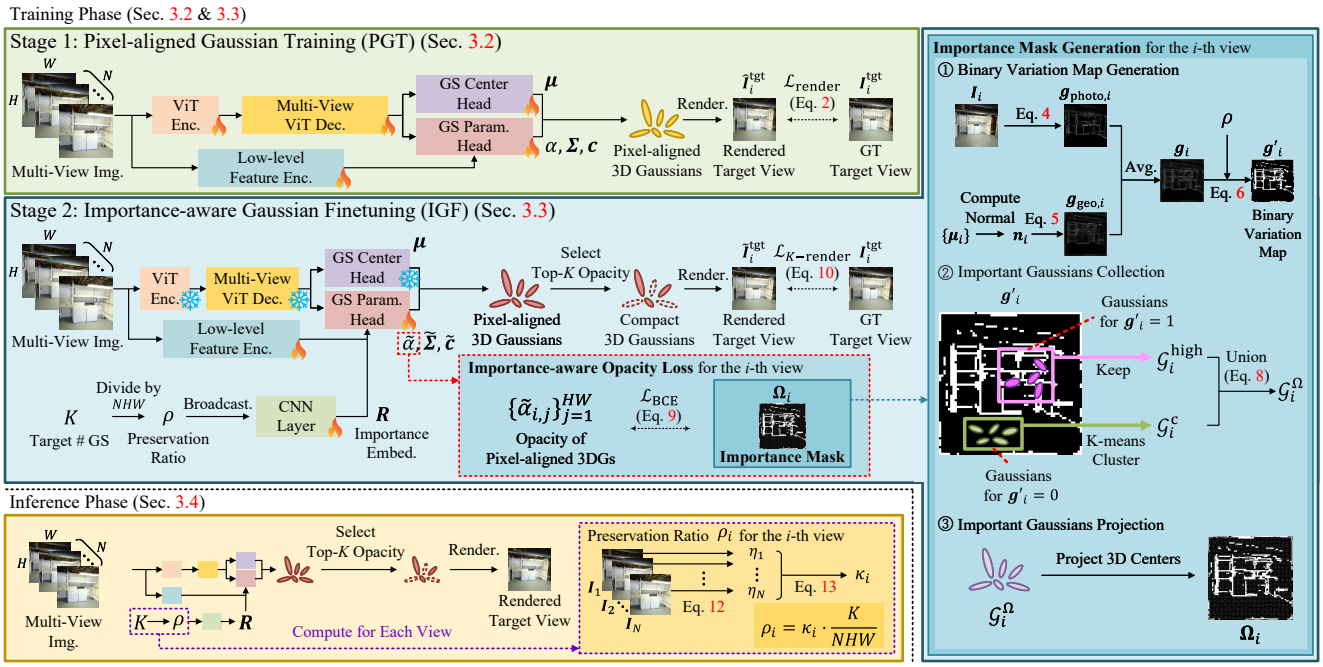


Figure 2. **Overview of EcoSplat.** EcoSplat is trained in two stages: Pixel-aligned Gaussian Training (PGT) (Sec. 3.2) and Importance-aware Gaussian Finetuning (IGF) (Sec. 3.3). During IGF, the combination of the importance-aware opacity loss $\mathcal{L}_{\text{io}}$ and the Progressive Learning on Gaussian Compaction (PLGC) encourages EcoSplat to suppress the opacities of less important Gaussians. At inference, it adaptively satisfies an arbitrary user-specified primitive count and produces the optimal Gaussians in a feed-forward manner (Sec. 3.4).

methods cannot operate in a feed-forward manner.

Feed-forward NVS. To enable generalizable NVS, early methods learn generalizable neural radiance fields [9, 43, 56] or image-based rendering models [35, 41, 46], leveraging data-driven priors via transformer-based architectures. More recently, due to the superior rendering quality and efficiency of 3DGS, several feed-forward approaches [8, 11, 24, 27, 42, 50, 53, 57] have been proposed to directly generate 3D Gaussians from input images. These methods predict the parameters of a single 3D Gaussian per pixel in each input image to represent the target scene. PixelSplat [8] and Splatter Image [42] are pioneering works that predict 3D Gaussians directly from image features. MVSplat [11] constructs a cost volume via plane sweeping to capture cross-view feature correspondences, while DepthSplat [50] leverages pre-trained monocular depth features [51] to achieve more robust 3D Gaussian reconstructions. Several methods [21, 24, 27, 33, 53] have been studied for pose-free feed-forward NVS. However, all of the aforementioned feed-forward methods are constrained by their *pixel-aligned Gaussian* representations, which inevitably introduce redundant Gaussians, limiting their efficiency and scalability for dense multi-view scenarios.

Efficient Feed-forward NVS. To address the inefficiency of the above *pixel-aligned* feed-forward NVS methods, recent studies [26, 36, 47, 59, 62] have proposed strategies for Gaussian aggregation and pruning. AnySplat [26] and WorldMirror [36], built upon the VGGT [45] architecture, cluster Gaussians into fixed-size voxels via differentiable voxelization, but the voxel size is a sensitive hyperparameter that strongly affects rendering quality. GGN [59] models the inter-view relationships among Gaussian groups using a graph-based approach and prunes redundant Gaussians via pooling. However, this graph-based approach has been shown to lack sufficient robustness, which hinders its generalization across diverse scene geometries and dense multi-view inputs. Zpressor [47] also resolves the bottleneck of inter-view relationships using key-frame clustering. Long-LRM [62] utilizes a Mamba2 [13]-based architecture to compress long sequences of Gaussians through token merging and opacity-based pruning. Despite of their advancements in efficient rendering, the aforementioned approaches provide neither explicit control over the desired number of primitives nor a guaranteed, optimal trade-off between rendering fidelity and primitive count.

3. Proposed Method: EcoSplat

3.1. Overview of EcoSplat

Given N multi-view images $\{\boldsymbol{I}_i\}_{i=1}^{N}$ for a scene and a target number of Gaussians K, we aim to predict, in a feed-forward manner, a set $\mathcal{G} = \{\boldsymbol{G}_k\}_{k=1}^{K}$ of 3D Gaussian primitives [28] that maximizes the rendering quality. Our proposed EcoSplat controls the per-view number of predicted

3D Gaussian primitives by ranking and selecting the most informative primitives in each view, ensuring that the set of selected primitives meets the target primitive number K.

To accomplish this, we adopt a two-stage optimization process: (i) The first stage is Pixel-aligned Gaussian Training (PGT) (Sec. 3.2) where we train our model to predict pixel-aligned Gaussians, similar to prior works [8, 11]; (ii) The second stage is Importance-aware Gaussian Finetuning (IGF) (Sec. 3.3) where our model is further trained to predict the Gaussian parameters (opacities, 3D covariances, and colors) adaptively based on the target primitive number K, enabling effective ranking and selection of important Gaussians. Specifically, we inject K as a conditioning signal into our model. Adaptive to K, EcoSplat suppresses the opacities of less important Gaussians in the pixel-aligned set, allowing the most informative ones to be retained. This process ensures the final output can be obtained by directly preserving the Gaussian primitives with the K-highest opacity scores, which are called 'top-K Gaussians'. For inference (Sec. 3.4), we describe our procedure, which satisfies the target primitive number K by allocating per-view Gaussian counts based on the importance of each view to maximize the rendering quality.

3.2. Pixel-aligned Gaussian Training (PGT)

In the PGT stage, we train a pixel-aligned feed-forward 3DGS model whose architecture is built on the recent methods [24, 53]. Each view image $\boldsymbol{I}_i$ is tokenized into P tokens. These tokens are then processed independently by a shared ViT [14] encoder to produce encoded tokens. The encoded tokens from all views are then fed jointly to a ViT decoder composed of m decoder blocks. This decoder applies cross-attention to aggregate multi-view information into a unified scene representation. The output of the decoder for the i-th view consists of multiple decoded tokens $\{\boldsymbol{Z}_i^{(\ell)}\}_{\ell=1}^{m}$, where each $\boldsymbol{Z}_i^{(\ell)} \in \mathbb{R}^{P\times C_\ell}$, with C_ℓ denoting the per-token channel dimension, is extracted from the ℓ-th decoder block. These decoded tokens are fed into a Gaussian center head F_μ and a Gaussian parameter head F_ν, which predict the 3D centers and all remaining parameters of the pixel-aligned Gaussians for the i-th view, respectively, as:

$$\begin{aligned}\{\boldsymbol{\mu}_{i,j}\}_{j=1}^{HW} &= F_\mu(\{\boldsymbol{Z}_i^{(\ell)}\}_{\ell=1}^{m}),\\ \{[\alpha_{i,j};\boldsymbol{\Sigma}_{i,j};\boldsymbol{c}_{i,j}]\}_{j=1}^{HW} &= F_\nu(\{\boldsymbol{Z}_i^{(\ell)}\}_{\ell=1}^{m}, \psi(\boldsymbol{I}_i)),\end{aligned} \tag{1}$$

where $\boldsymbol{\mu}_{i,j}$, $\alpha_{i,j}$, $\boldsymbol{\Sigma}_{i,j}$, and $\boldsymbol{c}_{i,j}$ represent the 3D center, opacity, 3D covariance, and color, respectively, of the 3D Gaussian $\boldsymbol{G}_{i,j}$ corresponding to the j-th pixel in the i-th view. $\psi(\boldsymbol{I}_i)$ denotes a feature map extracted from the input image $\boldsymbol{I}_i$ via a shallow CNN layer ψ that acts as a low-level feature encoder. Using the predicted pixel-aligned Gaussians, derived from all N input images, we render a target novel view image $\hat{\boldsymbol{I}}^{\text{tgt}}$ via rasterization with its corresponding camera. To train our model, we compute the rendering loss as the sum of the MSE and LPIPS [58] losses, averaged across all N^{tgt} target novel views as:

$$\mathcal{L}_{\text{render}} = \frac{1}{N^{\text{tgt}}}\sum_{p=1}^{N^{\text{tgt}}} \mathcal{L}_{\text{MSE}}(\boldsymbol{I}_p^{\text{tgt}}, \hat{\boldsymbol{I}}_p^{\text{tgt}}) + 0.05\cdot\mathcal{L}_{\text{LPIPS}}(\boldsymbol{I}_p^{\text{tgt}}, \hat{\boldsymbol{I}}_p^{\text{tgt}}). \tag{2}$$

3.3. Importance-aware Gaussian Finetuning (IGF)

Our EcoSplat mitigates the redundancy of pixel-aligned prediction, where the total number of generated primitives grows linearly with both the number of input views and the image resolution, leading to inefficient rendering. In the IGF stage, we aim to make our model aware of the target primitive number K, enabling it to adaptively predict the parameters of the pixel-aligned 3D Gaussians. For this, we finetune the Gaussian parameter head F_ν to adaptively suppress the opacities $\{\alpha_{i,j}\}$ of less important primitives, while simultaneously optimizing $\{\boldsymbol{\Sigma}_{i,j}\}$ and $\{\boldsymbol{c}_{i,j}\}$ to maximize rendering quality. We inject a learnable importance embedding $\boldsymbol{R}_i \in \mathbb{R}^{H\times W\times C}$, which is derived from the target primitive number K, into the intermediate feature of F_ν. Then, F_ν predicts the adjusted per-view pixel-aligned 3D Gaussian parameters as:

$$\{[\tilde{\alpha}_{i,j};\tilde{\boldsymbol{\Sigma}}_{i,j};\tilde{\boldsymbol{c}}_{i,j}]\}_{j=1}^{HW} = F_\nu(\{\boldsymbol{Z}_i^{(\ell)}\}_{\ell=1}^{m}, \psi(\boldsymbol{I}_i), \boldsymbol{R}_i), \tag{3}$$

where $\tilde{(\cdot)}$ denotes the adjusted Gaussian parameters. To predict $\boldsymbol{R}_i$, we compute the preservation ratio ρ_i from K as $\rho_i = \frac{K}{NHW}$ and broadcast ρ_i to an $H\times W$ tensor, and feed it into a shallow CNN layer. Note that ρ_i is set to a uniform value for all N views during training, while it is computed individually for each view during inference to maximize the rendering quality (Sec. 3.4). In the IGF stage, the ViT encoders/decoders and the Gaussian center head F_μ are kept frozen as shown in Fig. 2.

Importance Mask Generation. To adaptively finetune F_ν with respect to K, we aim to generate a pseudo ground truth (pseudo-GT) 2D importance mask Ω_i for each view. This Ω_i can be inferred from the set $\mathcal{G}_i^{\Omega}$ of important 3D Gaussians that should be retained. The importance mask generation process is divided into three steps, as follows:

1) Binary Variation Map Generation: First, we compute a variation map $\boldsymbol{g}_i \in \mathbb{R}^{H\times W}$ for $\boldsymbol{I}_i$ that captures both photometric and geometric complexities at each pixel. The photometric variation map $\boldsymbol{g}_{\text{photo},i} \in \mathbb{R}^{H\times W}$ is computed as the gradient magnitude of $\boldsymbol{I}_i$:

$$\boldsymbol{g}_{\text{photo},i} = \sqrt{\|\nabla_x \boldsymbol{I}_i\|_2^2 + \|\nabla_y \boldsymbol{I}_i\|_2^2}, \tag{4}$$

where $\|\cdot\|_2$ denotes the L2-norm. To compute the geometric variation map $\boldsymbol{g}_{\text{geo},i} \in \mathbb{R}^{H\times W}$, we obtain a depth map from the predicted means $\{\boldsymbol{\mu}_{i,j}\}_{j=1}^{HW}$ of the pixel-aligned 3D Gaussians and compute its corresponding normal map $\boldsymbol{n}_i$. Then, $\boldsymbol{g}_{\text{geo},i}$ is computed from $\boldsymbol{n}_i$ as:

$$\boldsymbol{g}_{\text{geo},i} = \sqrt{\|\nabla_x \boldsymbol{n}_i\|_2^2 + \|\nabla_y \boldsymbol{n}_i\|_2^2}. \tag{5}$$

The final variation map $\boldsymbol{g}_i$ is the average of the two variation maps, $\boldsymbol{g}_i = (\boldsymbol{g}_{\text{photo},i} + \boldsymbol{g}_{\text{geo},i})/2$. We then binarize $\boldsymbol{g}_i$, where $\boldsymbol{g}_{i,j}$ is the variation value at the j-th pixel location, to obtain a binary variation map $\boldsymbol{g}'_i$ as:

$$\boldsymbol{g}'_{i,j} = \begin{cases} 1, & \text{if } \boldsymbol{g}_{i,j} > \epsilon_i, \\ 0, & \text{otherwise}, \end{cases} \quad \text{where} \quad \epsilon_i = Q_{\rho_i}(\{\boldsymbol{g}_{i,j}\}_{j=1}^{HW}), \tag{6}$$

where $Q_{\rho_i}(\cdot)$ denotes the ρ_i-th quantile. Here, $\boldsymbol{g}'_{i,j} = 1$ indicates high-variation pixels and 0 othewise.

2) Important Gaussians Collection: Using the obtained binary variation map $\boldsymbol{g}'_i$, we select the set $\mathcal{G}_i^{\Omega}$ of important 3D Gaussians. We partition the predicted pixel-aligned 3D Gaussians from $\boldsymbol{I}_i$ using $\boldsymbol{g}'_i$ into two sets of 3D Gaussians:

$$\mathcal{G}_i^{\text{high}} = \{\boldsymbol{G}_{i,j} \mid \boldsymbol{g}'_{i,j} = 1\}, \quad \mathcal{G}_i^{\text{low}} = \{\boldsymbol{G}_{i,j} \mid \boldsymbol{g}'_{i,j} = 0\}. \tag{7}$$

Intuitively, high-variation pixels should be represented by individual primitives to preserve fine reconstruction details. Therefore, we fully include $\mathcal{G}_i^{\text{high}}$ within the set of important 3D Gaussians $\mathcal{G}_i^{\Omega}$, as $\mathcal{G}_i^{\Omega} \supseteq \mathcal{G}_i^{\text{high}}$. Next, we reduce the redundancy in the low-variation set $\mathcal{G}_i^{\text{low}}$ by merging its Gaussians using single-step K-means clustering. We first partition the input image $\boldsymbol{I}_i$ into non-overlapping 4×4 patches. Within the low-variation set $\mathcal{G}_i^{\text{low}}$, we designate the Gaussian associated with the first (top-left) pixel of each patch as a *key* Gaussian. The centers of these key Gaussians serve as cluster centroids. We then run single-step K-means clustering over $\mathcal{G}_i^{\text{low}}$ to compute the final compact set $\mathcal{G}_i^{\text{c}}$. This procedure reduces redundancy while preserving local structure, yielding a smaller, faithful set of Gaussians for subsequent processing. The final set $\mathcal{G}_i^{\Omega}$ is computed as:

$$\mathcal{G}_i^{\Omega} \triangleq \mathcal{G}_i^{\text{high}} \cup \mathcal{G}_i^{\text{c}}. \tag{8}$$

3) Important Gaussians Projection: The per-view importance mask Ω_i is generated by projecting the 3D centers of all Gaussians in $\mathcal{G}_i^{\Omega}$ onto the i-th image plane. This results in a binary mask where pixels intersected by any projected center are set to 1, and all other pixels are set to 0.

Loss Function. The model needs to learn suppressing the opacity of less important Gaussians while retaining those of more important Gaussians. For this, we introduce an importance-aware opacity loss $\mathcal{L}_{\text{io}}$ that encourages $\{\tilde{\alpha}_{i,j}\}_{j=1}^{HW}$ to match the importance mask Ω_i using a binary cross-entropy (BCE) loss as:

$$\mathcal{L}_{\text{io}} = \lambda_{\text{io}} \cdot \frac{1}{NHW} \sum_{i=1}^{N} \sum_{j=1}^{HW} \mathcal{L}_{\text{BCE}}\big(\Omega_{i,j},\, \tilde{\alpha}_{i,j}\big). \tag{9}$$

Finally, unlike the PGT stage, we only use the top-K Gaussians to render a target novel view image $\tilde{\boldsymbol{I}}^{\text{tgt}}$ via rasterization. The rendering loss is then computed as:

$$\mathcal{L}_{K\text{-render}} = \tfrac{1}{N^{\text{tgt}}} \sum_{p=1}^{N^{\text{tgt}}} \mathcal{L}_{\text{MSE}}(\boldsymbol{I}_p^{\text{tgt}}, \tilde{\boldsymbol{I}}_p^{\text{tgt}}) + 0.05 \cdot \mathcal{L}_{\text{LPIPS}}(\boldsymbol{I}_p^{\text{tgt}}, \tilde{\boldsymbol{I}}_p^{\text{tgt}}). \tag{10}$$

The total loss is computed as $\mathcal{L} = \mathcal{L}_{\text{io}} + \mathcal{L}_{K\text{-render}}$. Based on this total loss, our model is trained to predict an optimal set of primitives that maximizes rendering quality while adhering to the target primitive number K.

Progressive Learning on Gaussian Compaction (PLGC). It is important to stably train our model such that it can be robust to meet a wide range of target primitive numbers K. A naive approach for this is to randomly select K values over the entire training process. However, we found that this can lead to unstable training. To mitigate this, we propose a progressive learning strategy that expands the sampling range of K depending on the current training iteration. Specifically, we define the sampling interval as $[K_{\text{min}}, K_{\text{max}}]$, where the upper bound is fixed to $K_{\text{max}} = 0.95 \cdot NHW$. K_{min} is gradually annealed from $0.85 \cdot NHW$ down to $0.05 \cdot NHW$ according to:

$$K_{\text{min}} = \max\big(0.85 - \lambda_{\text{decay}} \lfloor t/S \rfloor,\, 0.05\big) \cdot NHW, \tag{11}$$

where t denotes the current iteration, S is the decay interval, and λ_{decay} is the decay rate. At each iteration, K is randomly sampled from the interval $[K_{\text{min}}, K_{\text{max}}]$. This sampling strategy allows the model to gradually adapt to more aggressive pruning, ensuring both stability and effectiveness across a wide range of target primitive numbers.

3.4. Inference

During inference, to predict the target K Gaussians, we allocate different numbers of primitives to each view, as some views contain more informative Gaussians than others. As the images with large portions of complex details tend to contain more informative primitives for scene details, more primitives are necessitated for such images. In this regard, we incorporate a view importance factor κ_i as the proportion of high-frequency components to indicate the amount of scene details in the i-th view. Thus, the view-specific preservation ratio ρ_i can be adaptively determined based on the view importance factor κ_i as $\rho_i = \kappa_i \rho$, where $\rho = \frac{K}{NHW}$. This adaptive allocation allows our EcoSplat to focus resources on the views that are more critical for reconstruction quality, while reducing redundancy in less informative views. Following MoBGS [6], we quantify the high-frequency components of each image $\boldsymbol{I}_i$ by defining a high-frequency score η_i as the ratio of spectral magnitude in the high-frequency region to the total spectral magnitude of the i-th image. We apply a 2D Discrete Fourier Transform (DFT) to $\boldsymbol{I}_i$ and shift the resulting spectrum to center the low-frequency components, yielding the shifted DFT $\tilde{\mathcal{F}}(\boldsymbol{I}_i)$. The high-frequency score η_i is then computed as:

$$\eta_i = 1 - \frac{\sum_{\boldsymbol{\xi} \in \Lambda} E_i(\boldsymbol{\xi})}{\sum_{\boldsymbol{\xi}} E_i(\boldsymbol{\xi})}, \quad \text{where} \quad E_i = |\tilde{\mathcal{F}}(\boldsymbol{I}_i)|, \tag{12}$$

where $\boldsymbol{\xi} = (w_x, w_y)$ denotes a 2D frequency index vector with horizontal and vertical frequencies of w_x and w_y, and

Methods	5% Gaussians			10% Gaussians			40% Gaussians			70% Gaussians		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
GGN [59]	N/A	N/A	N/A	N/A	N/A	N/A	15.86	0.580	0.408	15.71	0.577	0.408
AnySplat [26]	8.08	0.229	0.593	10.37	0.363	0.513	19.66	0.677	0.248	21.85	0.723	0.177
WorldMirror [36]	8.09	0.220	0.632	10.20	0.349	0.552	19.66	0.692	0.277	22.11	0.744	0.196
MVSplat [11] + [15]	5.82	0.053	0.717	6.41	0.085	0.696	8.71	0.224	0.598	9.26	0.245	0.581
MVSplat [11] + [20]	5.86	0.051	0.699	6.66	0.096	0.678	9.00	0.232	0.590	9.26	0.245	0.580
SPFSplat [24] + [15]	7.44	0.231	0.624	8.23	0.280	0.584	11.77	0.514	0.410	15.32	0.671	0.280
SPFSplat [24] + [20]	6.48	0.167	0.589	7.61	0.256	0.536	12.01	0.522	0.378	16.72	0.697	0.257
EcoSplat (Ours)	**24.72**	**0.822**	**0.183**	**25.00**	**0.831**	**0.171**	**25.11**	**0.835**	**0.164**	**25.00**	**0.832**	**0.166**

Table 1. **Quantitative comparison of NVS under controlled numbers of primitives on the RE10K dataset [61] with 24 input views.** **Red** and Blue indicate the best and second-best performances, respectively. 'N/A' denotes that GGN [59] cannot achieve configurations with primitive counts below 15% of the total number of pixel-aligned Gaussians due to its limited control over the primitive count.

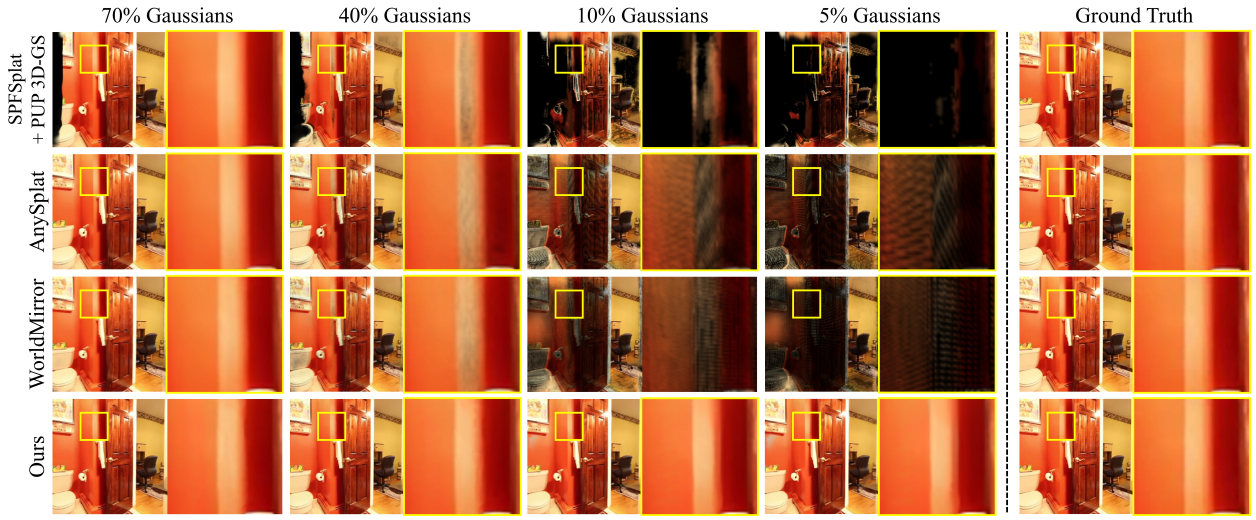


Figure 3. **Visual comparison of NVS under controlled numbers of primitives on the RE10K dataset [61] with 24 input views.**

Λ is the centered square region of side length s that captures the low-frequency portion of $\tilde{\mathcal{F}}(\boldsymbol{I}_i)$. Then, we compute the view-wise importance factor as:

$$\kappa_i = N \cdot \Psi_i, \quad \text{where} \quad \Psi_i = \frac{e^{\eta_i/T}}{\sum_{q=1}^{N} e^{\eta_q/T}}. \tag{13}$$

With temperature T, this softmax formulation amplifies differences among the high-frequency scores η_i across views. We then set $\rho_i = \kappa_i \rho$, thereby ensuring that the average preservation ratio over all input images is equal to ρ.

4. Experiments

Implementation Details. All details regarding our implementation are provided in the *Supplementary*.

Datasets. We train our EcoSplat model on the large-scale RealEstate10K (RE10K) dataset [61], which contains over 10 million frames collected from approximately 10K YouTube real estate videos. To assess our model's rendering quality, we use the RE10K test split and the ACID dataset [34] for in-domain and cross-domain generalization, respectively. The ACID dataset [34] consists of thousands of aerial drone YouTube videos depicting diverse coastal and natural scenes. Following prior works [11], we train and evaluate the models on 256×256 images.

Evaluation Protocol. We follow the evaluation protocol of NoPoSplat [53]. To evaluate model performance with a large number of input views, we first select challenging sequences from NoPoSplat characterized by large spatial gaps (0.05%-0.3% image overlap). We then densify these sparse sequences by randomly sampling intermediate views to meet our desired numbers of input views (e.g., 16, 20, 24 views). Following prior works [24, 26, 50, 53, 59], we quantitatively evaluate NVS quality using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [58]. The final number of Gaussians is also reported as an additional metric to highlight EcoSplat's robustness when rendering for various numbers of primitives.

Methods	16 Views				20 Views				24 Views			
	PSNR↑	SSIM↑	LPIPS↓	# GS↓	PSNR↑	SSIM↑	LPIPS↓	# GS↓	PSNR↑	SSIM↑	LPIPS↓	# GS↓
MVSplat [11]	14.88	0.463	0.435	1048K	14.87	0.462	0.437	1310K	14.86	0.461	0.440	1573K
DepthSplat [50]	20.05	0.752	0.258	1048K	19.53	0.733	0.274	1310K	19.18	0.718	0.286	1573K
NoPoSplat [53]	20.78	0.746	0.248	1048K	20.74	0.745	0.250	1310K	20.70	0.743	0.252	1573K
SPFSplat [24]	25.09	**0.841**	**0.141**	1048K	24.97	**0.838**	**0.142**	1310K	24.74	0.832	**0.145**	1573K
GGN [59]	15.82	0.578	0.409	425K	15.81	0.578	0.408	472K	15.80	0.578	0.408	512K
AnySplat [26]	21.73	0.720	0.177	887K	21.88	0.725	0.173	1078K	21.90	0.724	0.173	1259K
WorldMirror [36]	21.96	0.737	0.197	746K	22.07	0.741	0.195	891K	22.16	0.745	0.193	1020K
Ours$_{5\%}$	24.77	0.823	0.182	**52K**	24.78	0.825	0.181	**65K**	24.72	0.822	0.183	**78K**
Ours$_{40\%}$	**25.27**	0.839	0.161	419K	**25.21**	0.837	0.162	524K	**25.11**	**0.835**	0.164	629K

Table 2. **Quantitative comparison of NVS on the RE10K dataset [61] under various input-view settings.** We evaluate all models with 16, 20, and 24 input views. 'Ours$_{5\%}$' and 'Ours$_{40\%}$' refer to our EcoSplat model constrained to 5% and 40% of the total pixel-aligned Gaussians, respectively. We provide the inference FPS for all baselines in the *Supplementary*.

Methods	PSNR↑	SSIM↑	LPIPS↓	# GS↓
MVSplat [11]	18.09	0.500	0.379	1573K
DepthSplat [50]	19.78	0.688	0.289	1573K
NoPoSplat [53]	22.25	0.666	0.288	1573K
SPFSplat [24]	**24.40**	**0.720**	**0.222**	1573K
GGN [59]	18.06	0.553	0.389	475K
AnySplat [26]	21.96	0.619	0.258	1230K
WorldMirror [36]	22.34	0.658	0.273	1141K
Ours$_{5\%}$	23.87	0.689	0.282	**78K**
Ours$_{40\%}$	24.02	0.696	0.256	629K

Table 3. **Cross-dataset generalization with 24 input views.** We train all methods on the RE10K dataset [61] and evaluate their zero-shot performance on the ACID dataset [34]. We provide results for 16 and 20 views in the *Supplementary*.

4.1. Comparison with State-of-the-Art Methods

NVS under Controlled Number of Primitives. Table 1 and Fig. 3 present quantitative and qualitative comparisons of feed-forward 3DGS models under varying target 3D Gaussian counts, which are set to 5%, 10%, 40%, and 70% of the total pixel-aligned primitives (NHW). For this comparison, we use the RE10K dataset with 24 input views. We compare our EcoSplat with representative efficient feed-forward 3DGS methods, such as GGN [59], AnySplat [26], and WorldMirror [36]. Since these methods lack *explicit* control over the primitive counts at inference, we tune their relevant hyperparameters to *indirectly* control the primitive counts, such as similarity thresholds for GGN [59] and voxel sizes for AnySplat [26] and WorldMirror [36]. Furthermore, we compare our EcoSplat against the cascades of the SOTA pixel-aligned feed-forward 3DGS methods (SPFSplat [24] and MVSplat [11]) and the post-pruning methods [15, 20]. To preserve the feed-forward setting, we apply only the pruning step from [15, 20] to the pixel-aligned Gaussians predicted by SPFSplat [24] and MVSplat [11], with no additional optimization. As shown in Table 1, existing methods experience substantial degradation as the number of allowed primitives decreases. Due to its limited algorithmic design, GGN [59] cannot reduce its primitive counts below a certain minimum threshold. Voxel-based approaches, such as AnySplat [26] and WorldMirror [36], suffer from rendering artifacts at moderate pruning levels (e.g., 40%) and fail completely under aggressive pruning (e.g., 5%, 10%), as illustrated in Fig. 3. Although LightGaussian [15] and PUP 3D-GS [20] perform reasonably well for zero-shot pruning on *optimized*, per-scene 3DGS, they break down on feed-forward, pixel-aligned primitives. In these per-pixel representations, each Gaussian is tied to a single pixel, so pruning any primitive directly creates missing regions (Fig. 3). In contrast, EcoSplat maintains high rendering fidelity over a wide range of primitive counts, with only marginal degradation even under aggressive reduction of primitives. These results highlight EcoSplat's robustness of explicit primitive-count control.

NVS under Various Multi-view Settings. Table 2 shows the NVS performance comparison of our EcoSplat against recent SOTA feed-forward 3DGS methods [24, 26, 50, 53, 59, 62] across varying numbers of input views. For this comparison, we explicitly constrain our EcoSplat to extremely compact (5%) and moderate (40%) primitive ratios. As shown in Table 2, EcoSplat consistently achieves high rendering qualities across all input-view configurations while using substantially fewer Gaussian primitives. Moreover, even with a primitive budget **$\sim$7$\times$ smaller** than the most compact baseline, GGN [59], EcoSplat still achieves a **+9 dB** gain in PSNR and an LPIPS score that is **over 2$\times$ lower**. Compared to other efficient baselines [26, 36], EcoSplat delivers **2.5-3.5 dB** higher PSNR while requiring **over 10$\times$ fewer primitives**. These results demonstrate that EcoSplat combines superior rendering quality with explicit control over the primitive counts.

Cross-Dataset Generalization NVS. Table 3 shows the cross-dataset generalization performance of EcoSplat under multi-view settings, compared against recent SOTA feed-forward 3DGS methods [24, 26, 36, 50, 53, 59]. For this comparison, we use the RE10K dataset [61] for training

Variants	5% Gaussians			40% Gaussians		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
w/o PGT	22.93	0.778	0.221	23.70	0.806	0.184
w/o IGF	6.45	0.107	0.651	14.02	0.586	0.341
w/o $\mathcal{L}_{\text{io}}$	20.58	0.724	0.289	23.81	0.815	0.167
w/o PLGC	21.49	0.725	0.280	23.84	0.799	0.193
Ours	24.72	0.822	0.183	25.11	0.835	0.164

Table 4. **Ablation study on RE10K [61] with 24 input views.**

and the ACID dataset [34] for evaluation. EcoSplat consistently outperforms existing efficient feed-forward models and most of the pixel-aligned 3DGS methods except SPFSplat [24] in this setting. Note that EcoSplat underperforms SPFSplat with 0.38dB lower in PSNR, but uses substantially fewer primitives (only 40%).

4.2. Ablation Study

Training Strategy. We validate our two-stage training strategy by ablating each stage, removing PGT and IGF to obtain the variants 'w/o PGT' and 'w/o IGF', respectively, as shown in Table 4 and Fig. 4. Under a low primitive ratio (5%), 'w/o IGF' results in a drastic PSNR drop of over 18 dB and behaves similarly to the pixel-aligned feed-forward methods [11, 24] in Table 1. Likewise, 'w/o PGT' results in a 1.5 dB PSNR drop and visible rendering artifacts (Fig. 4), indicating the effectiveness of PGT.

IGF Components. We vaildate the core components of the IGF stage, $\mathcal{L}_{\text{io}}$ and the PLGC strategy in Table 4 and Fig. 4. Without $\mathcal{L}_{\text{io}}$ ('w/o $\mathcal{L}_{\text{io}}$'), our model improperly allocates top-K Gaussians relative to scene complexity, leading to under-reconstructed geometry and incomplete structural details (Fig. 4). This results in significant degradation in both structural (PSNR, SSIM) and perceptual (LPIPS) metrics, especially under aggressive pruning (5%). Similarly, without the PLGC strategy ('w/o PLGC'), our model results in unstable optimization and fails to effectively adjust Gaussian parameters, leading to distorted outputs (Fig. 4).

Importance-Aware Opacity. In Fig. 5, we visualize the opacity distributions predicted by EcoSplat for varying target primitive counts K on a given scene. When K is large ('70% Gaussians', orange), the model retains a substantial proportion of high-opacity primitives that capture fine geometric and appearance details. In contrast, when K is reduced to 5% (blue), EcoSplat effectively suppresses the opacities of low-importance primitives, pushing most of them toward the low-opacity region. Only a compact, highly informative subset maintains high opacity, corresponding to the most structurally and photometrically critical elements of the scene. This behavior illustrates that EcoSplat learns a robust importance ranking and dynamically adjusts Gaussian visibility according to the target K.

Figure 4. **Visual results of ablation study:** (a) the PGT stage; (b) the importance-aware opacity loss; and (c) the PLGC strategy.

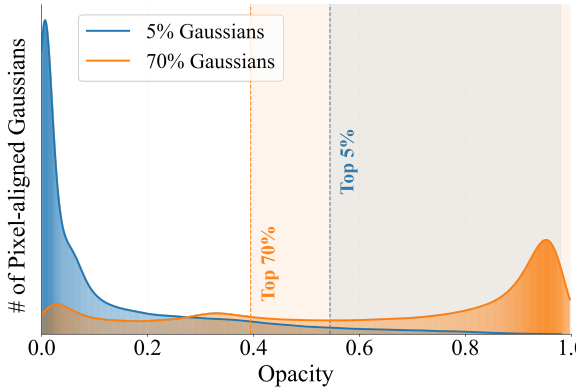


Figure 5. **Opacity distributions of pixel-aligned Gaussians across target primitive counts K.** The blue and orange curves show the distributions when the target primitive count is set to 5% and 70% of the total number of pixel-aligned Gaussians, respectively. EcoSplat adaptively predicts Gaussian opacities based on the primitive budget and selects the top-K Gaussians.

5. Conclusion

We present EcoSplat, the **first** feed-forward 3D Gaussian Splatting framework with **explicit control over the number of output Gaussian primitives** from multi-view images. EcoSplat enables flexible rendering on a wide range of end-user devices while maintaining high rendering quality. We validate the effectiveness of EcoSplat across extensive input-view settings and under various numbers of target primitives. Our experiments demonstrate that EcoSplat significantly outperforms SOTA methods using fewer primitives, and provides the freedom to dynamically control the primitive counts, offering an optimal trade-off between efficiency and rendering fidelity.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government [Ministry of Science and ICT (Information and Communications Technology)] (Project Number: RS-2022-00144444, Project Title: Deep Learning Based Visual Representational Learning and Rendering of Static and Dynamic Scenes, 100%).

References

[1] Milena T. Bagdasarian, Paul Knoll, Yi-Hsin Li, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 3dgs.zip: A survey on 3d gaussian splatting compression methods, 2025. 3

[2] 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. 1

[3] 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. 1

[4] Juan Luis Gonzalez Bello, Minh-Quan Viet Bui, and Munchurl Kim. Pronerf: Learning efficient projection-aware ray sampling for fine-grained implicit neural radiance fields. *IEEE Access*, 2024. 2

[5] Minh-Quan Viet Bui, Jongmin Park, Jihyong Oh, and Munchurl Kim. Moblurf: Motion deblurring neural radiance fields for blurry monocular video. *IEEE TPAMI*, 2025. 2

[6] Minh-Quan Viet Bui, Jongmin Park, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. Mobgs: Motion deblurring dynamic 3d gaussian splatting for blurry monocular video. In *AAAI*, 2026. 5

[7] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *CVPR*, 2023. 1, 2

[8] 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. 2, 3, 4

[9] 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. 3

[10] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 2

[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 *ECCV*, 2024. 2, 3, 4, 6, 7, 8, 1

[12] Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feedforward 3d gaussian splatting compression. In *ICLR*, 2025. 2, 3

[13] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024. 3

[14] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 4

[15] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compressison with 15x reduction and 200+ fps, 2024. 2, 6, 7, 1

[16] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 1, 2

[17] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 2

[18] Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A Efros, and Xiaolong Wang. Colmap-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20796–20805, 2024. 1

[19] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, 2021. 2

[20] Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In *CVPR*, 2025. 2, 6, 7, 1

[21] Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jiaolong Yang, Seungryong Kim, and Chong Luo. Unifying correspondence pose and nerf for generalized pose-free novel view synthesis. In *CVPR*, 2024. 3

[22] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf - efficient neural radiance fields. In *CVPR*, 2022. 2

[23] 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. 2

[24] Ranran Huang and Krystian Mikolajczyk. No pose at all: Self-supervised pose-free 3d gaussian splatting from sparse views. In *ICCV*, 2025. 2, 3, 4, 6, 7, 8, 1

[25] ISO/IEC JTC 1/SC 29/WG 2. Mpeg technical requirements - draft gaussian splat coding requirements, 2025. ISO/IEC JTC 1/SC 29/WG 2 N487, Geneva, Switzerland. 2

[26] 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. 2, 3, 6, 7, 1

[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. In *CVPR*, 2025. 2, 3

[28] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 2023. 1, 2, 3

[29] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. In *NeurIPS*, 2024. 1, 2

[30] Eugene Khvedchenya. Pytorch toolbelt. https://github.com/BloodAxe/pytorch-toolbelt, 2019. 1

[31] Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. In *CVPR*, 2025. 1

[32] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3d with mast3r. In *ECCV*, 2024. 1

[33] Hao Li, Yuanyuan Gao, Dingwen Zhang, Chenming Wu, Yalun Dai, Chen Zhao, Haocheng Feng, Errui Ding, Jingdong Wang, and Junwei Han. Ggrt: Towards generalizable 3d gaussians without pose priors in real-time. *arXiv preprint arXiv:2403.10147*, 2, 2024. 3

[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 *ICCV*, 2021. 2, 6, 7, 8, 1

[35] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *CVPR*, 2022. 3

[36] Yifan Liu, Zhiyuan Min, Zhenwei Wang, Junta Wu, Tengfei Wang, Yixuan Yuan, Yawei Luo, and Chunchao Guo. Worldmirror: Universal 3d world reconstruction with any-prior prompting. *arXiv preprint arXiv:2510.10726*, 2025. 3, 6, 7, 1, 2

[37] 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. 1, 2

[38] 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. 1, 2

[39] Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. Splinegs: Robust motion-adaptive spline for real-time dynamic 3d gaussians from monocular video. In *CVPR*, 2025. 1

[40] Zetian Song, Jiaye Fu, Jiaqi Zhang, Xiaohan Lu, Chuanmin Jia, Siwei Ma, and Wen Gao. Tinysplat: Feedforward approach for generating compact 3d scene representation. *IEEE TCSVT*, 2025. 2

[41] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *ECCV*, 2022. 3

[42] Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *CVPR*, 2024. 2, 3

[43] Mukund Varma T, Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. Is attention all that neRF needs? In *ICLR*, 2023. 3

[44] Chaoyang Wang, Ashkan Mirzaei, Vidit Goel, Willi Menapace, Aliaksandr Siarohin, Avalon Vinella, Michael Vasilkovsky, Ivan Skorokhodov, Vladislav Shakhrai, Sergey Korolev, Sergey Tulyakov, and Peter Wonka. 4real-video-v2: Fused view-time attention and feedforward reconstruction for 4d scene generation. In *NeurIPS*, 2025. 3

[45] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025. 3

[46] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 3

[47] Weijie Wang, Donny Y Chen, Zeyu Zhang, Duochao Shi, Akide Liu, and Bohan Zhuang. Zpressor: Bottleneck-aware compression for scalable feed-forward 3dgs. *arXiv preprint arXiv:2505.23734*, 2025. 2, 3

[48] 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. 1

[49] Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Yuri Makarov, Bingyi Kang, Xing Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. Spatialtrackerv2: 3d point tracking made easy. In *ICCV*, 2025. 3

[50] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *CVPR*, 2025. 2, 3, 6, 7, 1

[51] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *NeurIPS*, 2024. 3

[52] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024. 1

[53] 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. 2, 3, 4, 6, 7, 1

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

[55] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 1, 2

[56] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 3

[57] 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*, 2024. 2, 3

[58] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 4, 6

[59] Shengjun Zhang, Xin Fei, Fangfu Liu, Haixu Song, and Yueqi Duan. Gaussian graph network: Learning efficient and generalizable gaussian representations from multi-view images. *NeurIPS*, 2024. 2, 3, 6, 7, 1

[60] Lingzhe Zhao, Peng Wang, and Peidong Liu. Bad-gaussians: Bundle adjusted deblur gaussian splatting. In *ECCV*, 2024. 2

[61] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 2, 6, 7, 8, 1

[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. In *ICCV*, 2025. 2, 3, 7