F[4]Splat: Feed-Forward Predictive Densification for Feed-Forward 3D Gaussian Splatting

Injae Kim[1], Chaehyeon Kim[2], Minseong Bae[1], Minseok Joo[2], and Hyunwoo J. Kim[1†]

[1] KAIST, Daejeon, Republic of Korea
[2] Korea University, Seoul, Republic of Korea
https://mlvlab.github.io/F4Splat

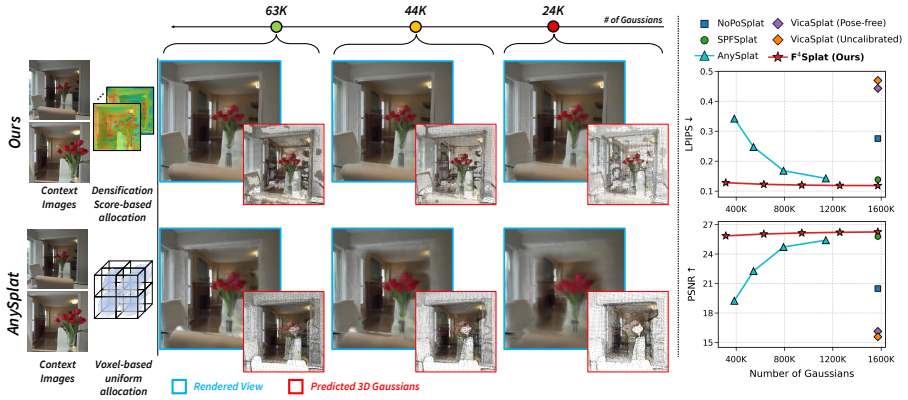
Abstract. Feed-forward 3D Gaussian Splatting methods enable single-pass reconstruction and real-time rendering. However, they typically adopt rigid pixel-to-Gaussian or voxel-to-Gaussian pipelines that uniformly allocate Gaussians, leading to redundant Gaussians across views. Moreover, they lack an effective mechanism to control the total number of Gaussians while maintaining reconstruction fidelity. To address these limitations, we present **F[4]Splat**, which performs ***F**eed-**F**orward predictive densification for **F**eed-**F**orward 3D Gaussian **Splat**ting*, introducing a densification-score-guided allocation strategy that adaptively distributes Gaussians according to spatial complexity and multi-view overlap. Our model predicts per-region densification scores to estimate the required Gaussian density and allows explicit control over the final Gaussian budget without retraining. This spatially adaptive allocation reduces redundancy in simple regions and minimizes duplicate Gaussians across overlapping views, producing compact yet high-quality 3D representations. Extensive experiments demonstrate that our model achieves superior novel-view synthesis performance compared to prior uncalibrated feed-forward methods, while using significantly fewer Gaussians.

Keywords: Feed-Forward 3D Gaussian Splatting · Compact 3DGS

1 Introduction

In modern computer vision, 3D scene reconstruction using deep learning has become the de facto standard. In particular, 3D Gaussian Splatting (3DGS) [25] has emerged as a highly efficient alternative to existing methodologies [14,38,40,44]. 3DGS represents scenes using an explicit set of 3D Gaussian primitives, enabling high-fidelity 3D reconstruction and real-time novel-view rendering. It incorporates adaptive density control (ADC), which periodically adds or removes Gaussians during optimization. These iterative updates assign a different number of Gaussians in each region, and through this adaptive assignment, the final 3DGS representation achieves high reconstruction fidelity with a relatively small number of Gaussians. However, the conventional 3DGS framework still inherits its key limitations shared by other optimization-based 3D reconstruction methods [14, 39, 40]. It requires costly per-scene iterative optimization and typically

† Corresponding author.



(a) Densification-score-guided allocation of F[4]Splat

(b) Rendering quality vs. Number of Gaussians

Fig. 1: Comparison of Gaussian allocation under different target Gaussian budgets. (a) Given the same target Gaussian budget, F[4]Splat allocates Gaussians non-uniformly using predicted densification scores, avoiding over-allocation in simple regions and concentrating primitives on fine-detail regions (e.g., flowers) to better preserve details even under a small budget, in contrast to voxel-based uniform allocation (AnySplat [23]). (b) Compared to pose-free and uncalibrated baselines, F[4]Splat achieves the best reconstruction fidelity (LPIPS & PSNR), while largely preserving the rendering quality as the number of Gaussians is reduced.

relies on densely captured input views with known camera parameters, which can be impractical in real-world scenarios. This has motivated feed-forward 3DGS methods [4, 8, 21, 23, 33, 55, 57], which are trained on large-scale datasets to build strong 3D priors. These frameworks can reconstruct a 3D scene from only a few input images through a single forward pass, preserving the real-time rendering capabilities of 3DGS while enabling generalization to unseen scenes. However, existing feed-forward 3DGS methods have a significant limitation in that they do not allocate Gaussians efficiently. This limitation stems from removing the iterative optimization process of conventional 3DGS, which also eliminates its periodic adaptive density control (ADC) that densifies Gaussians during training. Most works [4, 8, 21, 33, 55, 57, 63] adopt a pixel-to-Gaussian pipeline, which assigns Gaussians at the pixel level. This fixes the total number of Gaussians to the number of pixels in the entire image and prevents flexible adjustment of Gaussian positions, resulting in duplicated Gaussians across different views. AnySplat [23], which employs a voxel-to-Gaussian pipeline, can adjust the number of Gaussians by changing the voxel size, but this typically requires training a new model. Moreover, because it allocates Gaussians uniformly in space (i.e., assigning one Gaussian per voxel), it struggles to produce a high-quality and compact Gaussian representation under a limited Gaussian budget.

To address inefficient Gaussian allocation, we introduce **F[4]Splat**, a feed-forward network that performs *predictive densification* for 3D Gaussian Splatting from a set of uncalibrated images. Our approach predicts densification decisions in a single forward pass, treating Gaussian densification as a learnable

prediction problem within a unified feed-forward pipeline. Specifically, the network estimates a *densification score* that indicates whether additional Gaussians should be allocated to each region. By estimating both spatial complexity and multi-view overlap, the predicted densification score avoids over-allocation in simple regions and prevents duplicate Gaussians in areas covered by overlapping input images. This feed-forward densification strategy enables spatially adaptive allocation and yields compact Gaussian representations while maintaining competitive reconstruction fidelity. As illustrated in Fig. 1, this allows F[4]Splat to concentrate Gaussians on fine-detail regions while avoiding unnecessary allocation in simple regions, achieving higher rendering quality under the same Gaussian budget. Through extensive experiments, F[4]Splat achieves on-par or superior novel-view synthesis quality while using significantly fewer Gaussians than prior uncalibrated feed-forward methods that rely solely on image inputs.

The contributions of our work can be summarized as:

- **Gaussian-count controllable feed-forward 3DGS.** We propose F[4]Splat, a feed-forward framework that reconstructs 3D Gaussian Splatting representations from sparse, uncalibrated images while enabling explicit control over the final number of Gaussians through feed-forward predictive densification.
- **Densification-score-guided allocation for high fidelity under a limited budget.** We introduce a densification score that predicts where additional Gaussians should be allocated, enabling spatially adaptive Gaussian allocation without iterative optimization and maintaining high reconstruction fidelity even under a limited Gaussian budget.
- **State-of-the-art performance in the uncalibrated setting.** Extensive experiments show that F[4]Splat achieves on-par or superior novel-view synthesis quality while using significantly fewer Gaussians than prior uncalibrated feed-forward methods that rely solely on image inputs.

2 Related Work

3D Gaussian Splatting for Novel View Synthesis. NeRF [39] established a dominant paradigm for neural scene representation and sparked extensive subsequent research [1–3, 5, 14, 27, 37, 40, 45, 59], driving rapid progress in neural scene reconstruction. However, its per-ray volumetric rendering incurs high compute cost, motivating more efficient alternatives. 3DGS [25] mitigates this inefficiency by representing a scene with a set of 3D Gaussian primitives and rendering them through differentiable rasterization, enabling real-time rendering and faster optimization. To achieve high fidelity with a compact representation, it further employs *adaptive density control* (ADC), which periodically adds Gaussians in under-represented regions and prunes primitives with negligible contribution during iterative optimization. This has inspired a line of work on more compact 3DGS representations, spanning both refinements of the ADC strategy [10, 12, 16, 26, 28, 36, 47, 58, 61, 64] and various compaction and pruning methods [6, 11, 15, 30, 41, 43, 50, 56].

Despite the advantages of 3DGS, it still has several practical limitations. It typically assumes dozens to hundreds of diverse input views for stable reconstruction, which can be impractical in real-world scenarios. This dense-view requirement has been partly addressed by recent studies on *sparse-view* 3DGS [17, 29, 32, 54, 62, 67], which aim to reconstruct 3D scenes from only a few input images. Another major limitation is that 3DGS still requires iterative per-scene optimization, which remains a significant burden for practical deployment. To reduce this time-consuming optimization process, a variety of recent approaches [7, 13, 18, 49, 65] have sought to accelerate the original 3DGS optimization pipeline through more efficient rasterization, parallelization, and improved optimization strategies. Among these approaches, *feed-forward 3DGS* represents a particularly promising paradigm, as it amortizes iterative optimization into a single feed-forward pass, thereby enabling much faster 3D reconstruction.

Feed-Forward 3D Gaussian Splatting. Feed-forward 3DGS approaches [4, 8, 9, 19, 21, 23, 24, 33, 48, 53, 57, 63] have been proposed to alleviate the costly per-scene optimization of standard 3DGS. These methods are trained on large-scale datasets to learn strong priors, allowing them to predict 3D Gaussian representations in a single feed-forward pass without iterative optimization. Consequently, they can reconstruct from sparse views while enabling real-time rendering and generalization to unseen scenes. Early generalizable feed-forward 3DGS methods [4, 8, 53] typically assume calibrated multi-view inputs with known camera poses. Recent works [9, 48, 57, 63] relax this assumption by moving to pose-free settings. In addition, self-supervised pose-free approaches [19, 21, 24] further reduce reliance on pose annotations by learning from reconstruction consistency, with pose estimation integrated into the pipeline. More recently, uncalibrated formulations [23, 33] are enabling reconstruction without camera calibration.

Despite these advances in efficiency and robustness, existing feed-forward 3DGS methods largely rely on a *uniform* output parameterization, in which a fixed number of Gaussians is allocated per pixel or spatial unit. As a result, the total Gaussian count becomes tightly coupled with the input resolution, rather than being *adaptively* allocated according to scene complexity. This leads to redundant primitives in simple regions, while failing to sufficiently model geometrically complex regions, resulting in a suboptimal and non-compact representation under a limited Gaussian budget. In conventional optimization-based 3DGS, this issue has been mitigated through adaptive density control (ADC), which dynamically allocates Gaussians based on scene structure. However, such mechanisms rely on iterative per-scene optimization and are therefore not directly applicable to feed-forward pipelines. The recent feed-forward approach, AnySplat [23], is able to control the Gaussian count via voxel granularity. However, the allocation remains spatially uniform, and adapting to different budgets typically requires retraining, limiting flexibility and compactness. In contrast, we introduce Gaussian-count controllable feed-forward 3DGS, which predicts a budget-aware densification score that enables non-uniform and spatially adaptive Gaussian allocation. This yields a more compact 3D representation under a controllable Gaussian budget.

3 Method

We propose **F[4]Splat**, a feed-forward network that generates 3D Gaussian primitives [25] from an image collection via feed-forward predictive densification. Unlike prior feed-forward 3DGS methods that rely on uniform allocation, our method allows users to adjust the number of Gaussians on demand through spatially adaptive Gaussian allocation, making more effective use of the available Gaussian budget. In this section, we formulate the problem in Sec. 3.1. Next, Sec. 3.2 presents the overall framework, and Sec. 3.3 details the training pipeline.

3.1 Problem Formulation

Given N_{ctx} input context images $\{\mathbf{I}_i^{\text{ctx}}\}_{i=1}^{N_{\text{ctx}}}$, where $\mathbf{I}_i^{\text{ctx}} \in \mathbb{R}^{3\times H\times W}$, most prior feed-forward 3D Gaussian Splatting (3DGS) works [8, 21, 33, 55, 57, 63] uniformly allocate one Gaussian per pixel, resulting in a fixed number of Gaussians, $N_{\text{ctx}}HW$, to represent the scene. In contrast, our goal is to develop a feed-forward network $\mathbf{F}_\theta$ that enables control over the number of Gaussians. Specifically, $\mathbf{F}_\theta$ takes as input not only the context images but also a user-specified target Gaussian budget $\bar{N}_{\mathcal{G}}$, and predicts a set of 3D Gaussian primitives $\mathcal{G} = \{\mathbf{g}_g\}_{g=1}^{N_{\mathcal{G}}}$, where $N_{\mathcal{G}}$ does not exceed $\bar{N}_{\mathcal{G}}$, and camera parameters $\{\hat{\mathbf{P}}_i^{\text{ctx}}\}_{i=1}^{N_{\text{ctx}}}$:

$$\left(\{\mathbf{g}_g\}_{g=1}^{N_{\mathcal{G}}},\ \{\hat{\mathbf{P}}_i^{\text{ctx}}\}_{i=1}^{N_{\text{ctx}}}\right) = \mathbf{F}_\theta\left(\{\mathbf{I}_i^{\text{ctx}}\}_{i=1}^{N_{\text{ctx}}},\ \bar{N}_{\mathcal{G}}\right). \tag{1}$$

Each Gaussian primitive $\mathbf{g}_g \in \mathbb{R}^{d_{\mathbf{g}}}$ is parameterized by center $\boldsymbol{\mu}_g \in \mathbb{R}^3$, opacity $\sigma_g \in \mathbb{R}$, rotation in quaternion $\mathbf{q}_g \in \mathbb{R}^4$, scale $\mathbf{s}_g \in \mathbb{R}^3$, and spherical harmonics (SH) $\mathbf{h}_g \in \mathbb{R}^\nu$. Each camera parameter tuple is denoted as $\hat{\mathbf{P}}_i^{\text{ctx}} = (\hat{\mathbf{K}}_i^{\text{ctx}}, \hat{\mathbf{T}}_i^{\text{ctx}})$, where $\hat{\mathbf{K}}_i^{\text{ctx}} \in \mathbb{R}^{3\times 3}$ is the intrinsic matrix and $\hat{\mathbf{T}}_i^{\text{ctx}} \in \mathbb{R}^{4\times 4}$ is the camera-to-world pose. We use $\hat{f}_i^{\text{ctx}}$ to denote the focal length in $\hat{\mathbf{K}}_i^{\text{ctx}}$. Here, $\hat{(\cdot)}$ denotes a predicted value.

It is not enough to merely control the number of Gaussians; it is also important to use them efficiently to generate a high-quality scene representation. To this end, Gaussians should be allocated non-uniformly across the scene according to local characteristics, assigning more capacity to geometrically or visually complex regions. In the case of AnySplat [23], the number of Gaussians can be adjusted by changing the voxel size. However, due to the inherent limitation of uniformly assigning a single Gaussian primitive to each voxel, it represents the scene less faithfully even under the same Gaussian count. Uniform-Gaussian-allocation methods [4, 8, 21, 23, 33, 55, 57, 63] ignore the fact that different regions require different Gaussian densities, yielding redundant Gaussians in simple regions while under-allocating complex ones. Consequently, the Gaussian budget is not spent where it is most needed for faithful scene representation. To address this, we introduce a spatially adaptive Gaussian allocation framework that allocates Gaussians more effectively within a given budget.

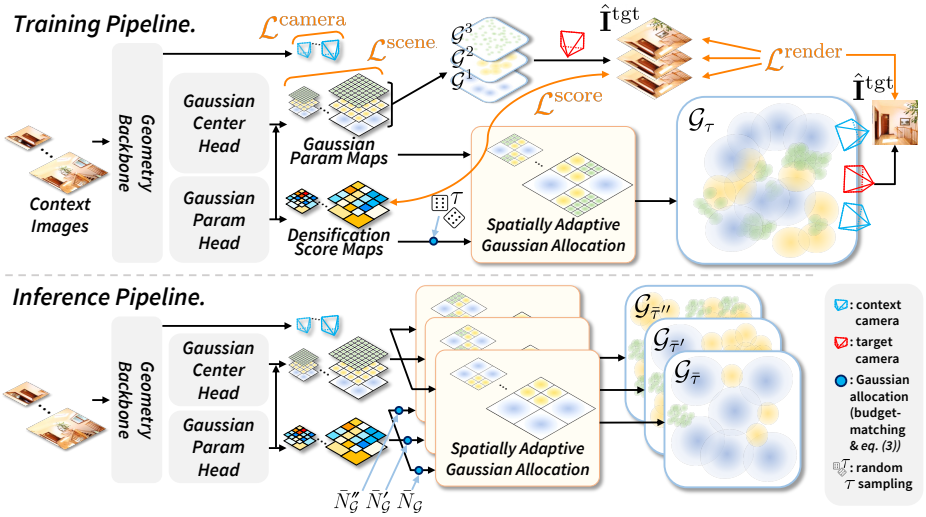


Fig. 2: Overview of F^{4}Splat. Given multi-view context images, the Geometry Backbone predicts camera parameters, multi-scale Gaussian parameter maps, and densification score maps. During training, camera and Gaussian parameters are jointly optimized with the camera loss $\mathcal{L}^{\text{camera}}$, rendering loss $\mathcal{L}^{\text{render}}$, and scene-scale regularization $\mathcal{L}^{\text{scene}}$. The predicted densification score maps are trained by score loss $\mathcal{L}^{\text{score}}$ derived from the backpropagation of rendering loss. The final representation $\mathcal{G}_\tau$, constructed using a randomly sampled threshold τ, is also supervised by the rendering loss $\mathcal{L}^{\text{render}}$. At inference time, given a target Gaussian budget $\bar{N}_\mathcal{G}$, the corresponding allocation threshold $\bar{\tau}$ is determined following the budget-allocation algorithm in Supplementary Sec. S1. By varying $\bar{N}_\mathcal{G}$, the same model produces a family of compact, high-fidelity, Gaussian-count controllable representations $\mathcal{G}_{\bar{\tau}}$ without retraining.

3.2 Spatially Adaptive Gaussian Allocation

As illustrated in Fig. 2, our framework consists of three parts: a *Geometry Backbone* that encodes geometric information from a multi-view image set and predicts camera parameters; *Gaussian Center Head* and *Gaussian Parameter Head*, which predict multi-scale Gaussian parameter maps along with densification score maps; and *Spatially Adaptive Gaussian Allocation* that effectively allocates the available Gaussian budget.

Geometry Backbone. To encode geometric information from a given image set $\{\mathbf{I}_i^{\text{ctx}}\}_{i=1}^{N_{\text{ctx}}}$, we adopt a geometric backbone following the structure of VGGT [51]. Each input image $\mathbf{I}_i^{\text{ctx}}$ is processed by a pretrained DINOv2 encoder [42] to extract patch tokens. The resulting image tokens $t_i^{\mathbf{I}}$ are concatenated with learnable camera tokens $t_i^{\mathbf{P}}$ and register tokens t_i^{r}. The reference view has its own learnable camera and register tokens, while the remaining views share their corresponding tokens. The combined tokens $\{[t_i^{\mathbf{I}}; t_i^{\mathbf{P}}; t_i^{\text{r}}]\}_{i=1}^{N_{\text{ctx}}}$ are then passed through alternating frame-wise and global self-attention layers, resulting in the encoded tokens $\{[\tilde{t}_i^{\mathbf{I}}; \tilde{t}_i^{\mathbf{P}}; \tilde{t}_i^{\text{r}}]\}_{i=1}^{N_{\text{ctx}}}$. The encoded camera tokens $\{\tilde{t}_i^{\mathbf{P}}\}_{i=1}^{N_{\text{ctx}}}$ are passed through four

additional self-attention layers, followed by a projection head to estimate camera parameters $\{\hat{\mathbf{P}}_i^{\text{ctx}}\}_{i=1}^{N_{\text{ctx}}}$.

Multi-Scale Prediction. To control the final number of Gaussians, multi-scale Gaussian parameter maps $\{\mathbf{G}_i^l\}_{l=1}^{L}$ and densification score maps $\{\hat{\mathbf{D}}_i^l\}_{l=1}^{L-1}$ are predicted from the encoded image tokens $\tilde{t}_i^{\mathbf{I}}$ encoded by the geometry backbone, where $\mathbf{G}_i^l \in \mathbb{R}^{d_{\mathbf{g}} \times H^l \times W^l}$ and $\hat{\mathbf{D}}_i^l \in \mathbb{R}^{H^l \times W^l}$. We modify a DPT-based decoder [46] to introduce two parallel heads, a *Gaussian Center Head* and a *Gaussian Parameter Head*. Before the final two layers, the decoded feature maps are bilinearly interpolated to the target resolution (H^l, W^l) at each level, and then the level-specific layers are applied. Each level-specific module consists of only two layers, enabling efficient multi-scale map prediction. The Gaussian center head predicts the Gaussian centers, and the Gaussian parameter head predicts the remaining Gaussian primitives along with the densification score maps. In the Gaussian parameter head, an RGB shortcut [57] is utilized before the level-specific layers. As the level l increases, the spatial resolution doubles at each step, $(H^{l+1}, W^{l+1}) = (2H^l, 2W^l)$. By exclusively selecting a scale level for each spatial region, we can control the final number of Gaussians $N_{\mathcal{G}}$. In the extreme case, selecting all regions from the coarsest level ($l=1$) yields $N_{\text{ctx}}H^1W^1$ Gaussians, while selecting all regions from the finest level ($l=L$) yields $N_{\text{ctx}}H^LW^L$ Gaussians. Therefore, $N_{\mathcal{G}}$ is bounded as:

$$N_{\text{ctx}}H^1W^1 \leq N_{\mathcal{G}} \leq N_{\text{ctx}}H^LW^L. \tag{2}$$

Spatially Adaptive Gaussian Allocation. Meanwhile, to represent a scene faithfully under a limited Gaussian budget, more Gaussians should be allocated to geometrically or photometrically complex regions. Additionally, redundant allocations to the same spatial locations across overlapping views should be minimized. If we can estimate how densely Gaussians should be in a given local space, we can allocate Gaussians more efficiently across the scene. To this end, we utilize a densification score map $\hat{\mathbf{D}}_i^l \in \mathbb{R}^{H^l \times W^l}$, which indicates how densely Gaussians should be placed in each spatial region. As shown in Fig. 3, thresholding the predicted densification scores yields spatially adaptive allocations, assigning more Gaussians to detail-rich regions while avoiding unnecessary allocation in simple or overlapping areas. More details on the computation of the densification map are provided in the following Sec. 3.3.

Using the densification score maps, we determine the appropriate representation level for each region via a simple thresholding rule. Starting from the coarsest level ($l=1$), if the densification score is higher than a given threshold τ, more Gaussians are allocated to that region from a higher-level Gaussian map. Ultimately, as illustrated in Fig. 2, Gaussians are selected such that allocations are non-overlapping across levels. The binary allocation masks $\mathcal{M}_{\tau,i}^l \in \{0,1\}^{H^l \times W^l}$, which indicate whether a particular location is allocated, can be computed as:

$$\mathcal{M}_{\tau,i}^l = \begin{cases} \mathbb{1}_{\{\hat{\mathbf{D}}_i^l < \tau\}} & \text{if } l = 1, \\ \mathbb{1}_{\{\hat{\mathbf{D}}_i^l < \tau\}} \odot \left(\mathbf{1} - \sum_{k=1}^{l-1} \text{Up}(\mathcal{M}_{\tau,i}^{l-k}; 2^k)\right) & \text{if } 1 < l < L, \\ \mathbf{1} - \sum_{k=1}^{l-1} \text{Up}(\mathcal{M}_{\tau,i}^{l-k}; 2^k) & \text{if } l = L, \end{cases} \tag{3}$$

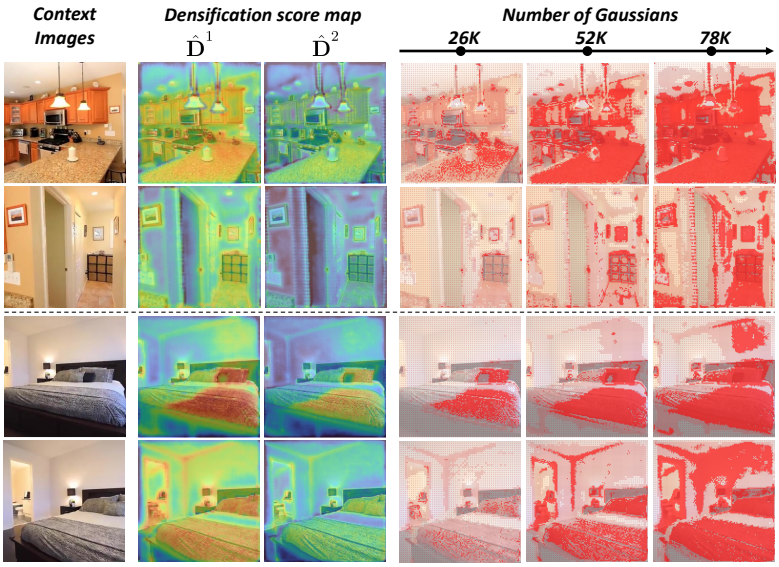


Fig. 3: Spatially adaptive Gaussian allocation. Given two context images (left), the model predicts densification score maps that estimate where additional Gaussian density is required. The red color indicates where more Gaussians are placed based on different fixed thresholds τ. In the top example, we can clearly observe that more Gaussians are allocated in complex regions. The example below demonstrates redundancy-aware allocation across overlapping views, avoiding unnecessary allocations.

where $\mathbb{1}_{\{\cdot\}}$ is an indicator function that outputs 1 when the condition is satisfied, $\mathbf{1}$ denotes a matrix of ones, $\odot$ is the element-wise product, and $\mathrm{Up}(\cdot; 2^k)$ denotes the nearest-neighbor upsampling with a scaling factor of 2^k. Using these masks, the final 3D Gaussian representation $\mathcal{G}_\tau = \{\mathbf{g}_g\}_{g=1}^{N_{\mathcal{G}_\tau}}$ is generated, where the total number of Gaussians $N_{\mathcal{G}_\tau} = \sum_{l,i} \|\mathcal{M}^l_{\tau,i}\|_1$. Given a target Gaussian-count budget $\bar{N}_\mathcal{G}$, we can compute the minimum threshold $\bar{\tau}$ that satisfies the target budget by a simple budget-matching algorithm, since the Gaussians are exclusively selected across levels. The computed threshold $\bar{\tau}$ guarantees that the final number of Gaussians $N_{\mathcal{G}_{\bar{\tau}}}$ satisfies the following conditions:

$$0 \leq \bar{N}_\mathcal{G} - N_{\mathcal{G}_{\bar{\tau}}} < 4^{L-1} - 1. \tag{4}$$

The budget-matching algorithm is provided in the Supplementary Sec. S1.

3.3 Training Strategy

Feed-Forward Predictive Densification. To allocate a limited number of Gaussians adaptively, the densification signal must satisfy two key properties. First, it should correlate with potential quality gain. It should allocate more Gaussians to under-represented regions and fewer to simple regions, so that representation quality tends to improve as the number of Gaussians increases. Second, it must be available at inference time without requiring iterative op-

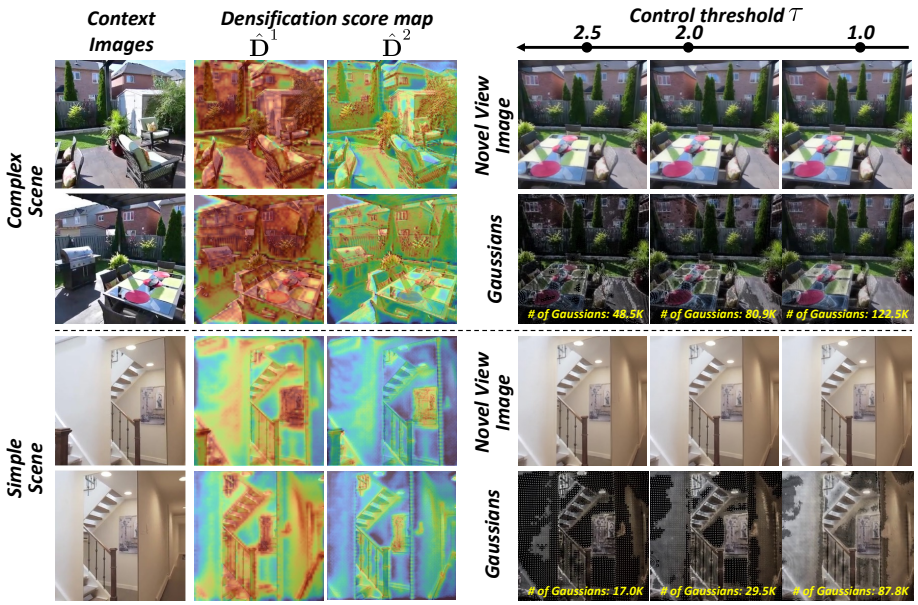


Fig. 4: Effect of the threshold τ across scene complexity. For a fixed τ, complex scenes yield higher densification scores and allocate more Gaussians, while simpler scenes allocate fewer.

timization. A desirable densification score must be computable using only the input images. To satisfy these conditions, we draw inspiration from the adaptive density control (ADC) strategy of standard 3DGS works [25, 58], which iteratively optimizes 3D Gaussian primitives.

These previous works [25,58] periodically densify 3D Gaussians during training. In AbsGS [58], for a set of predicted Gaussian primitives $\mathcal{G} = \{\mathbf{g}_g\}_{g=1}^{N_\mathcal{G}}$, whether to densify the Gaussian $\mathbf{g}_g$ is guided by the homodirectional target view-space positional gradient $\mathbf{v}_g$, which can be obtained by backpropagating the rendering loss. The rendering loss $\mathcal{L}^{\text{render}}$ is calculated between the predicted target image $\hat{\mathbf{I}}^{\text{tgt}}$ and the ground-truth target image $\mathbf{I}^{\text{tgt}}$. With this loss, we can calculate the homodirectional view-space positional gradient $\mathbf{v}_g$:

$$\mathbf{v}_g = \left(\sum_{j=1}^{m} \left| \frac{\partial \mathcal{L}_j^{\text{render}}}{\partial \bar{\boldsymbol{\mu}}_{g,x}} \right|, \sum_{j=1}^{m} \left| \frac{\partial \mathcal{L}_j^{\text{render}}}{\partial \bar{\boldsymbol{\mu}}_{g,y}} \right| \right). \tag{5}$$

Here $(\bar{\boldsymbol{\mu}}_{g,x}, \bar{\boldsymbol{\mu}}_{g,y})$ denotes the center of Gaussian $\mathbf{g}_g$ after projection onto the 2D image plane of the target view, $\mathcal{L}_j^{\text{render}}$ denotes the rendering loss computed by the j-th pixel of the image $\hat{\mathbf{I}}^{\text{tgt}}$, and m is the total number of pixels that Gaussian $\mathbf{g}_g$ participates in rendering of $\hat{\mathbf{I}}^{\text{tgt}}$. A large value of $\mathbf{v}_g$ indicates that the Gaussian $\mathbf{g}_g$ significantly affects the rendering loss, implying that the corresponding region is underrepresented. AbsGS [58] empirically shows that as-

signing more Gaussians based on the norm of this gradient improves the fidelity of the representation. Therefore, this gradient-based value is a suitable signal for our spatially adaptive Gaussian allocation pipeline. However, feed-forward 3DGS predicts 3D Gaussian primitives in a single forward pass without iterative scene optimization, and ground-truth images are not available at inference time. As a result, the gradient $\mathbf{v}_g$ cannot be utilized directly at inference time.

To overcome this limitation, we instead *learn to predict* $\hat{d}_g$ from the gradient $\mathbf{v}_g$. Here, $\hat{d}_g$ denotes the predicted densification score associated with the spatial region occupied by the Gaussian $\mathbf{g}_g$. During the training process, using the 3D Gaussian representation $\mathcal{G}$ predicted by input context images $\{\mathbf{I}_i^{\text{ctx}}\}_{i=1}^{N_{\text{ctx}}}$, we can compute the rendering loss $\mathcal{L}^{\text{render}}(\hat{\mathbf{I}}^{\text{tgt}}, \mathbf{I}^{\text{tgt}})$ and the homodirectional view-space positional gradient $\mathbf{v}_g$ for each Gaussian $\mathbf{g}_g \in \mathcal{G}$, as explained above. Based on the accumulated gradient $\mathbf{v}_g$, we define the supervision signal d_g for the densification as a log-scaled value of the ℓ_2 norm of the gradient: $d_g = \log\left(1 + 10^4 \cdot \|\mathbf{v}_g\|_2\right)$. Then, the predicted densification score $\hat{d}_g$ for $\mathbf{g}_g$ is trained to match d_g via the following ℓ_1 loss:

$$\mathcal{L}_{\mathcal{G}}^{\text{score}} = \mathbb{E}_{\mathbf{g}_g \in \mathcal{G}} \left[\|\hat{d}_g - d_g\|_1 \right]. \tag{6}$$

This gradient-based supervision also implicitly captures cross-view redundancy: regions already well represented by overlapping context views produce lower densification signals, suppressing duplicate Gaussian allocation, as visualized in Fig. 3. Furthermore, since the score is learned from gradient signals, it serves as an absolute and comparable criterion across different scenes, rather than a purely relative ranking within a single scene. Consequently, the threshold τ serves as a control knob for reconstruction fidelity, determining the level of detail preserved in the final representation. As shown in Fig. 4, under a fixed τ, a complex scene generally produces higher densification scores, resulting in higher Gaussian allocation. In contrast, a simple scene yields lower scores and consequently fewer Gaussians.

Training with Novel Views. Training the model by supervising the final 3D Gaussian representation using the input context views, rather than novel views, can lead to trivial reconstruction and potential overfitting, as the model may simply memorize the observed views instead of learning a representation that generalizes across viewpoints. To address this issue, unlike the previous feed-forward approach [23] that supervises reconstruction of input context views, we instead use a novel view as the target image during training. However, directly using the ground-truth camera parameters of the target view is problematic because the camera coordinate system predicted by the model may differ from the ground-truth coordinate frame. To resolve this mismatch, we align the ground-truth target camera into the predicted coordinate system before rendering the target view. Specifically, let $\mathbf{T}_{n_1}^{\text{ctx}}$ and $\mathbf{T}_{n_2}^{\text{ctx}}$ denote the ground-truth poses of the two context views closest to the target view, and $\hat{\mathbf{T}}_{n_1}^{\text{ctx}}$ and $\hat{\mathbf{T}}_{n_2}^{\text{ctx}}$ denote their corresponding predicted poses. We estimate a similarity transformation matrix $\mathbf{A} \in \text{Sim}(3)$ that maps poses from the ground-truth coordinate frame to the predicted one. The context view n_1 closest to the target view is used as the anchor

reference for rotation and translation alignment, while the scale is estimated from the relative translation between the two context views n_1 and n_2 closest to the target view. The ground-truth target pose $\mathbf{T}^{\text{tgt}}$ is then transformed using the matrix $\mathbf{A}$ to obtain the aligned target pose, $\hat{\mathbf{T}}^{\text{tgt}} = \mathbf{A}\mathbf{T}^{\text{tgt}}$. For the intrinsic parameters f, we adjust the focal length of the target view using the ratio between the predicted and ground-truth focal lengths of the nearest context view, $\hat{f}^{\text{tgt}} = \hat{f}^{\text{ctx}}_{n_1} / f^{\text{ctx}}_{n_1} \cdot f^{\text{tgt}}$. This simple yet carefully designed alignment procedure ensures that the aligned target view parameters $(\hat{\mathbf{T}}^{\text{tgt}}, \hat{\mathbf{K}}^{\text{tgt}})$ are geometrically consistent with the coordinate system predicted by the model, enabling stable and reliable novel view training.

Overall Training Pipeline. During training, we randomly sample a threshold τ and predict the corresponding Gaussian set $\mathcal{G}_\tau$, which is supervised using the rendering loss $\mathcal{L}^{\text{render}}$. In addition, we also apply the rendering loss to level-wise Gaussian representations $\mathcal{G}^l \in \mathbb{R}^{d_{\mathbf{g}} \times N_{\text{ctx}} H^l W^l}$, which are constructed using only the Gaussians predicted at each level l. The densification score loss $\mathcal{L}^{\text{score}}$ is computed using the densification scores derived from the multi-level Gaussian representations $\{\mathcal{G}^l\}_{l=1}^{L-1}$. The loss for camera parameter estimation, $\mathcal{L}^{\text{camera}}$, is learned in the same way as VGGT [51]. We further introduce a scene-scale regularization loss $\mathcal{L}^{\text{scene}}$, which normalizes the average distance of Gaussian centers from the origin to 1, *i.e.*, $\mathcal{L}^{\text{scene}} = \left| \frac{1}{|\mathcal{G}|} \sum_{\mathbf{g}_g \in \mathcal{G}} \|\boldsymbol{\mu}_g\|_2 - 1 \right|$. This regularization is applied independently to the representation of each level. Without this regularization, training can become unstable. Finally, the total training objective is defined as the weighted sum of $\mathcal{L}^{\text{render}}$, $\mathcal{L}^{\text{score}}$, $\mathcal{L}^{\text{camera}}$, and $\mathcal{L}^{\text{scene}}$.

4 Experiments

Datasets. We train F[4]Splat on the large-scale RealEstate10K (RE10K) [66], ACID [35] datasets, and DL3DV [34], following prior work [57] for train/test splits. For two-view evaluation, we adopt the same test split as prior feed-forward methods [4, 8, 33, 55, 57]. For multi-view evaluation, we use the scene categorization provided by NoPoSplat [57] and select input pairs with small overlap. Additional input views are sampled between the selected pair without duplication to match the target number of input views (8, 16, and 24 views).

Implementation Details. We use three levels of multi-resolution feature maps ($L = 3$) in all experiments and choose the finest level to match the input image resolution, $(H^L, W^L) = (H, W)$. We initialize our model from pretrained VGGT weights. We use a learning rate of 2×10^{-4} for most modules, while freezing the patch embedding weights of the geometry backbone and training the remaining backbone parameters with a smaller learning rate of 2×10^{-5}. For multi-view training, we train on RE10K. At each iteration, each GPU independently samples the number of input views from $\{2, 3, 4, 6, 12, 24\}$ and selects context images accordingly. We additionally sample the same number of target novel views for supervision. To keep the total number of images processed per iteration constant, we use a dynamic batch size that is inversely proportional to the number of context images. The multi-view model is trained for 15,000 iterations. For

Table 1: Novel view synthesis performance on RE10K [66] under different numbers of input views. We report LPIPS/SSIM/PSNR for 8, 16, and 24 input views. Best and second-best results are highlighted in **bold** and underlined, respectively. τ^+ and τ^- denote the high- and low-threshold variants, respectively.

	Method	8 views				16 views				24 views			
		#GS↓	LPIPS↓	SSIM↑	PSNR↑	#GS↓	LPIPS↓	SSIM↑	PSNR↑	#GS↓	LPIPS↓	SSIM↑	PSNR↑
Pose-Free	NoPoSplat	524K	0.213	0.756	22.31	1049K	0.252	0.713	21.11	1573K	0.275	0.691	20.49
	VicaSplat	524K	0.241	0.713	21.18	1049K	0.384	0.596	17.56	1573K	0.443	0.546	16.13
	SPFSplat	524K	0.142	0.849	25.66	1049K	0.137	0.855	25.88	1573K	0.139	0.853	25.78
Uncalibrated	VicaSplat	524K	0.258	0.686	20.77	1049K	0.417	0.556	16.78	1573K	0.470	0.517	15.58
	AnySplat	447K	0.167	0.819	24.07	820K	0.148	0.842	25.10	1142K	0.143	0.849	25.40
	F^{4}Splat$_{\tau^+}$	**105K**	0.140	0.851	25.35	**210K**	0.126	0.866	25.91	**315K**	0.122	0.870	26.01
	F^{4}Splat$_{\tau^-}$	447K	**0.128**	**0.863**	**25.73**	820K	**0.116**	**0.875**	**26.28**	1142K	**0.113**	**0.879**	**26.43**

Table 2: Generalization to unseen datasets. Models are trained on RE10K [66] and evaluated on the unseen ACID dataset [35]. We compare reconstruction quality under different numbers of input views (8, 16, and 24).

	Method	8 views				16 views				24 views			
		#GS↓	LPIPS↓	SSIM↑	PSNR↑	#GS↓	LPIPS↓	SSIM↑	PSNR↑	#GS↓	LPIPS↓	SSIM↑	PSNR↑
Pose-Free	NoPoSplat	524K	0.268	0.672	22.84	1049K	0.294	0.644	22.14	1573K	0.307	0.630	21.78
	VicaSplat	524K	0.271	0.656	22.57	1049K	0.327	0.609	21.20	1573K	0.356	0.581	20.37
	SPFSplat	524K	0.215	0.736	24.89	1049K	0.214	0.743	25.07	1573K	0.218	0.744	25.00
Uncalibrated	VicaSplat	524K	0.315	0.588	21.31	1049K	0.377	0.547	19.82	1573K	0.399	0.529	19.22
	AnySplat	481K	0.248	0.696	23.30	906K	0.236	0.720	23.88	1289K	0.234	0.727	24.04
	F^{4}Splat$_{\tau^+}$	**52K**	0.230	0.726	24.55	**105K**	0.215	0.745	25.02	**315K**	0.195	0.768	25.51
	F^{4}Splat$_{\tau^-}$	481K	**0.194**	**0.759**	**25.16**	906K	**0.184**	**0.775**	**25.56**	1289K	**0.181**	**0.780**	**25.70**

two-view training, separate models are trained on RE10K and ACID, following the training configuration of NoPoSplat [57]. Each model is trained for 18,750 iterations with a total batch size of 128. We set the weight of MSE and LPIPS loss as 1 and 0.05 for $\mathcal{L}^{\text{render}}$. For the final total loss, we fix the weight of $\mathcal{L}^{\text{render}}$, $\mathcal{L}^{\text{score}}$, $\mathcal{L}^{\text{camera}}$, and $\mathcal{L}^{\text{scene}}$ as 1.0, 10^{-4}, 10.0, and 10^{-2} respectively, in all experiments. All experiments are conducted on eight NVIDIA H200 GPUs, and each training run takes approximately 15 hours.

Baselines and Evaluation Metrics. For multi-view evaluation, we compare F^{4}Splat with recent feed-forward 3DGS methods under two settings: *uncalibrated* methods that take only images as input without camera poses or intrinsics, and *pose-free* methods that do not require camera poses but assume known intrinsics. For the uncalibrated setting, we compare against VicaSplat [33] and AnySplat [23]. For fair comparison, we retrain AnySplat under the same training setup as ours, which requires approximately 27 hours on eight NVIDIA H200 GPUs. VicaSplat uses the officially released multi-view pretrained weights on the RE10K dataset. For the pose-free setting, we compare against NoPoSplat [57], VicaSplat [33], and SPFSplat [21], using their officially released pretrained models. For two-view evaluation, we compare with pixelSplat [4], MVSplat [8], NoPoSplat [57], VicaSplat [33], and SPFSplat [21]. We also compare baselines that are not a feed-forward 3D Gaussian splatting method, including pixelNeRF [60], DUSt3R [52], MASt3R [31], and CoPoNeRF [20]. We evaluate the performance of novel view synthesis using three standard metrics: LPIPS, SSIM, and PSNR. We also report the number of final Gaussian primitives to enable joint comparison of rendering quality and Gaussian-count efficiency.

Table 3: Novel view synthesis performance comparison under 2-view setting. Results on ACID [35] dataset.

	Method	Average			
		#GS↓	LPIPS↓	SSIM↑	PSNR↑
Pose-Required	pixelNeRF	-	0.533	0.561	20.32
	pixelSplat	131K	0.195	0.779	25.82
	MVSplat	131K	0.196	0.773	25.51
Pose-Free	DUSt3R	-	0.447	0.411	16.29
	MASt3R	-	0.461	0.409	16.18
	CoPoNeRF	-	0.406	0.606	20.95
	NoPoSplat	131K	0.189	0.781	25.96
	VicaSplat	131K	0.201	0.757	25.44
	SPFSplat	131K	0.176	0.807	26.80
Uncalibrated	VicaSplat	131K	0.218	0.726	24.55
	F[4]Splat$_{\tau+}$	**26K**	0.201	0.772	25.75
	F[4]Splat$_{\tau-}$	131K	**0.178**	**0.794**	**26.26**

Table 4: Ablation Studies. Experiments are conducted with 24 input views, where the Gaussian budget is fixed to 20% of the maximum number of Gaussians. (a)-(b) compare different Gaussian allocation strategies, (c) removes the level-wise Gaussian supervision during training, and (d) removes the scene-scale regularization term.

	Variant	LPIPS↓	SSIM↑	PSNR↑
(a)	Rand. based allocation	0.194	0.828	24.68
(b)	Freq. based allocation	0.160	0.841	25.36
(c)	w/o level-wise GS train	0.192	0.813	24.25
(d)	w/o scene scale reg.	0.712	0.006	4.82
(e)	**Ours**	**0.143**	**0.854**	**25.47**

4.1 Experimental Results

Multi-view Reconstruction. As shown in Tab. 1 and Tab. 2, our method achieves competitive performance among uncalibrated baselines and remains competitive with pose-free and pose-required methods. When using a similar number of Gaussian primitives as the baselines, our approach achieves strong reconstruction performance. Notably, even when using only 10-28% of the Gaussian primitives, our method still maintains competitive results. These results suggest that our explicit density control enables a compact yet high-fidelity 3D Gaussian representation, leading to improved efficiency without sacrificing reconstruction quality.

Two-view Sparse Reconstruction. Tab. 3 demonstrates the novel view synthesis performance on the ACID dataset under the challenging two-view setting. Our method achieves superior performance compared to the uncalibrated baseline. Moreover, it remains competitive with both pose-required and pose-free approaches, including feed-forward 3DGS methods and other baselines, demonstrating robust performance even with only two sparse input views.

Qualitative Comparison. Fig. 5 shows qualitative comparisons on RE10K under different input-view settings. Our method produces sharper structures and more faithful scene details compared to existing approaches. Notably, even with substantially fewer Gaussian primitives (24-29%), our approach maintains high rendering quality while reducing blurring artifacts, resulting in visually more accurate renderings.

Generalization Under Larger-Scale Data. While Tab. 2 evaluates cross-dataset generalization from RE10K to ACID, we further examine whether the advantage of F[4]Splat persists under a larger-scale training setting. Specifically, we train on RE10K+DL3DV and evaluate zero-shot performance on DTU under the two-view setting, a held-out 3D reconstruction benchmark. As shown in Tab. 5, F[4]Splat outperforms NoPoSplat across all metrics, indicating that our predictive densification strategy remains effective when scaling the training data.

Fig. 5: Qualitative comparisons of novel view synthesis on RE10K [66]. The bottom-right corner of each image shows the PSNR of the rendered view and the number of Gaussian primitives used for scene reconstruction. Our method achieves high-quality rendering even with substantially fewer primitives, while consistently outperforming competing approaches.

4.2 Ablation Studies and Analysis

Densification-Score-Guided Allocation. We validate the effectiveness of our learned densification score for Gaussian allocation by comparing it with two alternative strategies, (a) and (b) in Tab. 4. For (a) *random-based allocation*, we replace the learned densification score with a uniform score, which leads to random Gaussian allocation. For (b) *frequency-based allocation*, we compute a heuristic frequency score by resizing the input image to each scale level and applying a Sobel filter to emphasize high-frequency (edge) regions. Both alternatives underperform, demonstrating that the learned densification score provides a more effective signal for allocating Gaussians than either random selection or simple frequency-based heuristics. Qualitatively, Fig. 3 further illustrates the behavior of the learned densification score. The predicted maps of the top example assign higher scores to structurally and visually complex regions (e.g., object boundaries, textured areas, and fine details), indicating where increased Gaussian density is likely to be beneficial. Moreover, as in the example below, overlapping areas observed from multiple context images tend to have lower scores, reflecting redundancy-awareness across views. This encourages allocating Gaussians to regions that need more capacity while avoiding redundancy in well-covered areas, improving efficiency and fidelity under a fixed budget.

Level-Wise Gaussian Supervision. Next, we examine the impact of level-wise Gaussian supervision during training. If we supervise only the final Gaussian set $\mathcal{G}_\tau$, the sampled threshold τ causes some scale-level Gaussians to be excluded from training in each iteration, resulting in less stable optimization. As shown in Tab. 4, removing level-wise supervision clearly degrades performance (c), confirming the benefit of supervising multi-scale Gaussians during training.

Table 5: Generalization on DTU under larger-scale training. Models are trained on RE10K [66] and DL3DV [34], and evaluated on the DTU [22] dataset.

Method	#GS↓	PSNR↑	SSIM↑	LPIPS↓
NoPoSplat	131K	17.51	0.570	0.304
F[4]Splat	**26K**	**19.76**	**0.647**	**0.271**

Table 6: Effect of our VGGT-prior training strategy. Our strategy leverages the VGGT prior more stably than the AnySplat-like alternative, enabling stable novel-view training while reducing training time.

Strategy	LPIPS↓	SSIM↑	PSNR↑	Train. Time↓
AnySplat-like	0.244	0.711	20.59	5H 35M
Ours	**0.171**	**0.829**	**24.62**	**3H 03M**

Scene-Scale Regularization. Finally, we ablate the scene-scale regularization loss in Tab. 4 (d). Without this term, training becomes highly unstable, and the model fails to learn a meaningful reconstruction, leading to training failure. In the uncalibrated setting, where the model must jointly predict camera parameters and scene geometry, this simple regularizer is crucial for stable optimization.

VGGT-prior training strategy. Tab. 6 analyzes how to effectively leverage the VGGT prior for feed-forward 3DGS training. The AnySplat-like alternative performs input-view training with frozen VGGT distillation, whereas our strategy enables stable novel-view training through aligned target-camera supervision and scene-scale regularization. As shown in Tab. 6, our strategy achieves better reconstruction quality while reducing training time.

5 Conclusion

We present F[4]Splat, a feed-forward 3DGS framework that reconstructs compact representations from sparse, uncalibrated inputs. By introducing a feed-forward densification through a densification-score-guided allocation strategy, our method adaptively distributes Gaussians according to spatial complexity and multi-view overlap, enabling explicit control over the final Gaussian budget without retraining. By allocating more primitives to informative regions while avoiding redundancy in simple or overlapping areas, F[4]Splat produces compact Gaussian representations while achieving high reconstruction fidelity. Experiments demonstrate that our approach achieves competitive or superior novel-view synthesis performance compared to prior feed-forward methods while requiring significantly fewer Gaussians, highlighting the effectiveness of spatially adaptive Gaussian allocation for efficient feed-forward 3DGS reconstruction.

Acknowledgments. This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00443251, Accurate and Safe Multimodal, Multilingual Personalized AI Tutors, 30%), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2023R1A2C2005373, 30%), the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2024-00457882, AI Research Lab Project, 30%), and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (RS-2025-25442149, LG AI STAR Talent Development Program for Leading Large-Scale Generative AI Models in the Physical AI Domain, 10%).

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: ICCV (2021)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: CVPR (2022)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. In: ICCV (2023)
4. Charatan, D., Li, S.L., Tagliasacchi, A., Sitzmann, V.: pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In: CVPR (2024)
5. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: ECCV (2022)
6. Chen, Y., Wu, Q., Lin, W., Harandi, M., Cai, J.: Hac: Hash-grid assisted context for 3d gaussian splatting compression. In: ECCV (2024)
7. Chen, Y., Jiang, J., Jiang, K., Tang, X., Li, Z., Liu, X., Nie, Y.: Dashgaussian: Optimizing 3d gaussian splatting in 200 seconds. In: CVPR (2025)
8. Chen, Y., Xu, H., Zheng, C., Zhuang, B., Pollefeys, M., Geiger, A., Cham, T.J., Cai, J.: Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In: ECCV (2024)
9. Chen, Z., Yang, J., Yang, H.: Pref3r: Pose-free feed-forward 3d gaussian splatting from variable-length image sequence. arXiv preprint arXiv:2411.16877 (2024)
10. Cheng, K., Long, X., Yang, K., Yao, Y., Yin, W., Ma, Y., Wang, W., Chen, X.: Gaussianpro: 3d gaussian splatting with progressive propagation. In: ICML (2024)
11. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z., et al.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. NeurIPS (2024)
12. Fang, G., Wang, B.: Mini-splatting: Representing scenes with a constrained number of gaussians. In: ECCV (2024)
13. Feng, G., Chen, S., Fu, R., Liao, Z., Wang, Y., Liu, T., Hu, B., Xu, L., Pei, Z., Li, H., et al.: Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. In: CVPR (2025)
14. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022)
15. Girish, S., Gupta, K., Shrivastava, A.: Eagles: Efficient accelerated 3d gaussians with lightweight encodings. In: ECCV (2024)
16. Grubert, G., Barthel, F.T., Hilsmann, A., Eisert, P.: Improving adaptive density control for 3d gaussian splatting. In: VISIGRAPP (2025)
17. He, Z., Xiao, Z., Chan, K.C., Zuo, Y., Xiao, J., Lam, K.M.: See in detail: Enhancing sparse-view 3d gaussian splatting with local depth and semantic regularization. In: ICASSP (2025)
18. Höllein, L., Božič, A., Zollhöfer, M., Nießner, M.: 3dgs-lm: Faster gaussian-splatting optimization with levenberg-marquardt. In: ICCV (2025)
19. Hong, S., Jung, J., Shin, H., Han, J., Yang, J., Luo, C., Kim, S.: Pf3plat: Pose-free feed-forward 3d gaussian splatting. In: ICML (2025)
20. Hong, S., Jung, J., Shin, H., Yang, J., Kim, S., Luo, C.: Unifying correspondence, pose and nerf for pose-free novel view synthesis from stereo pairs. arXiv preprint arXiv:2312.07246 (2023)
21. Huang, R., Mikolajczyk, K.: No pose at all: Self-supervised pose-free 3d gaussian splatting from sparse views. In: ICCV (2025)

22. Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., Aanæs, H.: Large scale multi-view stereopsis evaluation. In: ICCV (2014)
23. Jiang, L., Mao, Y., Xu, L., Lu, T., Ren, K., Jin, Y., Xu, X., Yu, M., Pang, J., Zhao, F., et al.: Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. ACM Trans. Graphics (2025)
24. Kang, G., Yoo, J., Park, J., Nam, S., Im, H., Shin, S., Kim, S., Park, E.: Selfsplat: Pose-free and 3d prior-free generalizable 3d gaussian splatting. In: CVPR (2025)
25. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graphics (2023)
26. Kheradmand, S., Rebain, D., Sharma, G., Sun, W., Tseng, Y.C., Isack, H., Kar, A., Tagliasacchi, A., Yi, K.M.: 3d gaussian splatting as markov chain monte carlo. NeurIPS (2024)
27. Kim, I., Choi, M., Kim, H.J.: Up-nerf: Unconstrained pose prior-free neural radiance field. In: NeurIPS (2023)
28. Kim, S., Lee, K., Lee, Y.: Color-cued efficient densification method for 3d gaussian splatting. In: CVPR Workshops (2024)
29. Kong, H., Yang, X., Wang, X.: Generative sparse-view gaussian splatting. In: CVPR (2025)
30. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. In: CVPR (2024)
31. Leroy, V., Cabon, Y., Revaud, J.: Grounding image matching in 3d with mast3r. In: ECCV (2024)
32. Li, J., Zhang, J., Bai, X., Zheng, J., Ning, X., Zhou, J., Gu, L.: Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In: CVPR (2024)
33. Li, Z., Dong, C., Chen, Y., Huang, Z., Liu, P.: Vicasplat: A single run is all you need for 3d gaussian splatting and camera estimation from unposed video frames. arXiv preprint arXiv:2503.10286 (2025)
34. Ling, L., Sheng, Y., Tu, Z., Zhao, W., Xin, C., Wan, K., Yu, L., Guo, Q., Yu, Z., Lu, Y., et al.: Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In: CVPR (2024)
35. Liu, A., Tucker, R., Jampani, V., Makadia, A., Snavely, N., Kanazawa, A.: Infinite nature: Perpetual view generation of natural scenes from a single image. In: ICCV (2021)
36. Mallick, S.S., Goel, R., Kerbl, B., Steinberger, M., Carrasco, F.V., De La Torre, F.: Taming 3dgs: High-quality radiance fields with limited resources. In: SIGGRAPH Asia (2024)
37. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: CVPR (2021)
38. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: CVPR (2019)
39. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
40. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graphics (2022)
41. Niedermayr, S., Stumpfegger, J., Westermann, R.: Compressed 3d gaussian splatting for accelerated novel view synthesis. In: CVPR (2024)

42. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
43. Papantonakis, P., Kopanas, G., Kerbl, B., Lanvin, A., Drettakis, G.: Reducing the memory footprint of 3d gaussian splatting. Proc. ACM Comput. Graphics Comput. Syst. (2024)
44. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019)
45. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: ICCV (2021)
46. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: ICCV (2021)
47. Rota Bulò, S., Porzi, L., Kontschieder, P.: Revising densification in gaussian splatting. In: ECCV (2024)
48. Smart, B., Zheng, C., Laina, I., Prisacariu, V.A.: Splatt3r: Zero-shot gaussian splatting from uncalibrated image pairs. arXiv preprint arXiv:2408.13912 (2024)
49. Wang, C., Ma, G., Xue, Y., Lao, Y.: Faster and better 3d splatting via group training. In: ICCV (2025)
50. Wang, H., Zhu, H., He, T., Feng, R., Deng, J., Bian, J., Chen, Z.: End-to-end rate-distortion optimized 3d gaussian representation. In: ECCV (2024)
51. Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., Novotny, D.: Vggt: Visual geometry grounded transformer. In: CVPR (2025)
52. Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B., Revaud, J.: Dust3r: Geometric 3d vision made easy. In: CVPR (2024)
53. Wang, Y., Huang, T., Chen, H., Lee, G.H.: Freesplat: Generalizable 3d gaussian splatting towards free view synthesis of indoor scenes. NeurIPS (2024)
54. Xiong, H., Muttukuru, S., Upadhyay, R., Chari, P., Kadambi, A.: Sparsegs: Real-time 360° sparse view synthesis using gaussian splatting. arXiv preprint arXiv:2312.00206 (2023)
55. Xu, H., Peng, S., Wang, F., Blum, H., Barath, D., Geiger, A., Pollefeys, M.: Depthsplat: Connecting gaussian splatting and depth. In: CVPR (2025)
56. Yang, R., Zhu, Z., Jiang, Z., Ye, B., Chen, X., Zhang, Y., Chen, Y., Zhao, J., Zhao, H.: Spectrally pruned gaussian fields with neural compensation. arXiv preprint arXiv:2405.00676 (2024)
57. Ye, B., Liu, S., Xu, H., Li, X., Pollefeys, M., Yang, M.H., Peng, S.: No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. In: ICLR (2025)
58. Ye, Z., Li, W., Liu, S., Qiao, P., Dou, Y.: Absgs: Recovering fine details in 3d gaussian splatting. In: ACMMM (2024)
59. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: ICCV (2021)
60. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: CVPR (2021)
61. Zhang, J., Zhan, F., Xu, M., Lu, S., Xing, E.: Fregs: 3d gaussian splatting with progressive frequency regularization. In: CVPR (2024)
62. Zhang, J., Li, J., Yu, X., Huang, L., Gu, L., Zheng, J., Bai, X.: Cor-gs: sparse-view 3d gaussian splatting via co-regularization. In: ECCV (2024)
63. Zhang, S., Wang, J., Xu, Y., Xue, N., Rupprecht, C., Zhou, X., Shen, Y., Wetzstein, G.: Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In: CVPR (2025)

64. Zhang, Z., Hu, W., Lao, Y., He, T., Zhao, H.: Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting. In: ECCV (2024)
65. Zhao, H., Weng, H., Lu, D., Li, A., Li, J., Panda, A., Xie, S.: On scaling up 3d gaussian splatting training. In: ICLR (2025)
66. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. ACM Trans. Graphics (2018)
67. Zhu, Z., Fan, Z., Jiang, Y., Wang, Z.: Fsgs: Real-time few-shot view synthesis using gaussian splatting. In: ECCV (2024)