# Beyond Existance: Fulfill 3D Reconstructed Scenes with Pseudo Details

Yifei Gao\* Jun Huang\*
yilei.jin123@gmail.com

Lei Wang†

Ruiting Dai

Jun Cheng

## Abstract

The emergence of 3D Gaussian Splatting (3D-GS) has significantly advanced 3D reconstruction by providing high fidelity and fast training speeds across various scenarios. While recent efforts have mainly focused on improving model structures to compress data volume or reduce artifacts during zoom-in and zoom-out operations, they often overlook an underlying issue: training sampling deficiency. In zoomed-in views, Gaussian primitives can appear unregulated and distorted due to their dilation limitations and the insufficient availability of scale-specific training samples. Consequently, incorporating pseudo-details that ensure the completeness and alignment of the scene becomes essential. In this paper, we introduce a new training method that integrates diffusion models and multi-scale training using pseudo-ground-truth data. This approach not only notably mitigates the dilation and zoomed-in artifacts but also enriches reconstructed scenes with precise details out of existing scenarios. Our method achieves state-of-the-art performance across various benchmarks and extends the capabilities of 3D reconstruction beyond training datasets.

## 1. Introduction

The advent of 3D Gaussian Splatting (3D-GS) [25] has inspired significant advancements in 3D reconstruction, enhancing a variety of applications such as VR interactions [21, 71], drivable human avatars [48, 53, 83, 85], and navigation through large-scale urban scenes [36, 75, 84]. Furthermore, the implementation of 3D-GS in 4D reconstruction [8, 16, 68], splashing effects [11], style transformation [33], and object segmentation and editing [77] have emerged, revealing significant commercial potential.

Despite the promising progress of 3D Gaussian Splatting (3D-GS), its relatively straightforward optimization strategies have led to persistent issues such as artifacts in novel-view renderings, redundancy, and limited rendering speed. To tackle these problems, Mip-Splatting [80] utilizes a 3D smoothing filter and a 2D Mip filter to mitigate zoom-in artifacts and zoom-out dilation observed in 3D-GS. Similarly, Mipmap-GS [28] integrates multi-scale pseudo-

ground truth training to enhance performance. On another front, Scaffold-GS [36] introduces spatial offsets to improve the structure of Gaussians and incorporates a Multi-Layer Perceptron (MLP) for implicit color prediction and generalization. Efforts in compression and quantization [9, 17, 42] aim to reduce the redundancy of Gaussians while largely maintaining their rendering potential. Octree-GS [50] employs an octree structure to enhance Level-of-Detail (LOD) reconstruction, achieving both high rendering quality and volumetric compactness. In contrast to methods that focus on improving modeling techniques, Bootstrap-GS [15] concentrates on enhancing the sampling process of training data. It enhances training results by bootstrapping novelview renderings from already trained 3D-GS and reintegrating them into the training sets while maintaining multi-view consistency.

Despite the progress and contributions of Bootstrap-GS, its training strategies tend to be coarse and time-consuming, and its training process remains precarious when striving to fully alleviate artifacts. Furthermore, while contemporary methods can largely recover scenes in their original views, they struggle to handle zoomed-in renderings using 3D-GS, with most exhibiting significant artifacts in zoomed-in views, as shown in Figure 1. Although the filtering approach in Mip-Splatting helps alleviate these issues, it often results in blurred outputs that lack high-frequency details.

Building upon prior research [15, 50, 80], we introduce a novel training strategy that enhances fine details using upscale diffusion models while ensuring consistency between high-frequency zoomed-in details and low-frequency training data. Additionally, we have developed a refined bootstrapping pipeline to regulate time consumption and minimize fluctuations during training. To validate the effectiveness of our methods, we apply them across multiple frameworks and evaluate their performance on diverse benchmarks.

In summary, our contributions are as follows:

- We theoretically demonstrate the feasibility of generating details from a zoomed-in perspective using upscale diffusion models, and achieve excellent results in practical applications.
- · By integrating upscaling diffusion models during train-

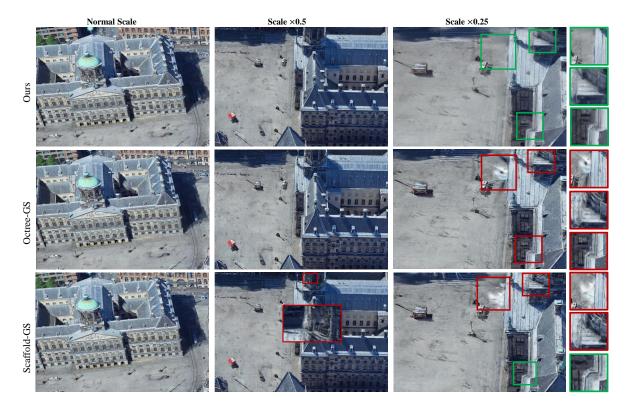


Figure 1. **Rendering comparisons.** While most methods render images faithfully at standard scales, their zoomed-in views exhibit significant artifacts that are not apparent at normal magnifications.

ing, our method generates high-frequency, out-of-scene details that remain consistent with the training datasets in standard views, while simultaneously mitigating artifacts in zoomed-in views.

 We fully exploit bootstrapping techniques and refine its pipeline to achieve state-of-the-art performance across various tasks based on different frameworks, while ensuring a stable and efficient training process.

#### 2. Related Work

**Image Distortion of Upscaling:** Scaling adjustments during the training of 3D-GS are achieved by modifying the training camera scales and correspondingly adjusting the resolutions of their ground truth images. For amplified scales, the image resolution is increased using interpolation [6, 28, 80]. However, when upscaling an image, increasing the sampling rate during interpolation by considering more surrounding pixels cannot overcome the fundamental limit on the highest frequency that can be sampled.

In practice, to satisfy the constraint of Nyquist-Shannon Sampling Theorem [44, 55] when reconstructing a signal from discrete samples, a low-pass or anti-aliasing filter is applied before sampling. This filter eliminates any frequency components above half of the sampling rate and attenuates high-frequency content that could lead to aliasing.

Consequently, after filtering, the highest signal frequency obtainable from a given image is less than half of the highest original frequency.

Traditional upscaling methods cannot restore high-frequency details from filtered signals. In practical applications of upscaling, inadequate sampling can lead to image artifacts such as dilation and aliasing. These issues become particularly significant in regions rich in detail when the upscaling ratio exceeds 2, resulting in noticeable distortions.

Mulfi-view Generation by Diffusion Model: Diffusion models [19, 43, 56–58] are latent variable models using Markov chain to recover the original data distribution  $\mathbf{x}_0$  from an initial random noise. Training is performed by optimizing the standard variational bound on the negative log-likelihood, and the training objective simplifies to [19]:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\epsilon} \left[ \left\| \epsilon - \epsilon_{\theta} (\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t) \right\|^2 \right]$$
(1)

where  $\epsilon_{\theta}$  is a learned noise prediction function,  $\bar{\alpha}_t$  is the schedule factor, t is the current time step, and  $\epsilon$  is a random constant.

Diffusion models have been frequently employed as priors in training optimization for 3D reconstruction tasks, particularly in object reconstruction [7, 35]. Score Jacobian Chaining is proposed to lift pretrained 2D diffusion model

for 3D object generation [66]. In DreamFusion [47], Score Distillation Sampling was introduced to eliminate the need for explicit sampling from diffusion models during training, but only in the latent space. However, DreamGaussian [61] demonstrated that this method is ineffective for generating fine-grained details, where exact sampling and further fine-tuning of diffusion models become necessary. Prior to the emergence of Bootstrap-GS, diffusion models were seldom used as priors in large scene reconstruction, primarily because the multi-view alignment of the generated views by diffusion poses significant implementation difficulties.

#### 3. Preliminaries

## 3.1. Bootstrap-GS

Due to the deficiency in scenario sampling, a bootstrapping technique is employed in [15] for scenario integration based on 3D-GS. For a brief introduction to 3D-GS, please refer to our **Supplementary Material**. A novel-view image **I** synthesized by the trained 3D-GS model [25] can be considered a "partially degraded" version of the ground truth image **I**'. To approximate the ground truth, a diffusion model (denoiser)  $\epsilon_{\theta}$  with learnable parameters  $\theta$  is utilized to perform image-to-image regeneration on the synthesized image, generating a differential  $\delta_i$  such that:  $\mathbf{I}' \approx \delta_i + \mathbf{I}$ . The differential  $\delta_i$  can be represented as a sum over reverse diffusion processes based on Eq. 1:

$$\delta_i = \sum_{t \in T} \epsilon_{\theta} \left( \sqrt{\bar{\alpha}_t} \, \mathbf{x}_t (\mathbf{I}, \sqrt{1 - \bar{\alpha}_t} \, \boldsymbol{\epsilon}) \right), \tag{2}$$

where  $\mathbf{x}_t$  is a function of  $\mathbf{I}$  and  $\boldsymbol{\epsilon}$ , T is a set of pre-specified time schedule [19, 57].

To enhance multi-view consistency in regenerated images, Bootstrap-GS employs multiple renderings focused on the same region to perform average sampling. Suitable thresholds are set to ensure faithful cloning and splatting of Gaussians. Given a set of novel-view renderings  $\mathbf{I}_n$  and their regenerated counterparts  $\mathbf{I}_r$ , the bootstrapping loss  $\mathcal{L}_b$  is defined as the average  $\mathcal{L}_1$  loss over each pair of images. Specifically,  $\mathcal{L}_b = \frac{\lambda_{\text{boot}}}{N} \sum_{i \in N} \mathcal{L}_b^i$ , where  $\mathcal{L}_b^i = \|\mathbf{I}_n^i - \mathbf{I}_r^i\|$ ,  $\lambda_{\text{boot}}$  is a scalar weighting factor, and N is the total number of variants. By combining this average loss with an appropriate  $\lambda_{\text{boot}}$ , the bootstrapping loss  $\mathcal{L}_b$  increases the average sampling steps for the regenerated parts, leading to more faithful re-rendering. For more details, please refer to [15].

#### 4. Methods

In this section, we first elucidate the theoretical basis of using upscale diffusion models and then illustrate our practical implementations.

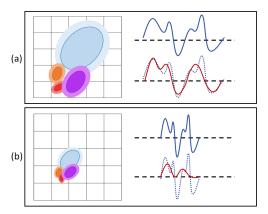


Figure 2. Signal visualization of the rendering process of Gaussian primitives. (a) presents the zoomed-in view, while (b) displays the normal view. The left side is the Gaussians and the right side is their corresponding signals. The original signal is represented by the blue curve, and the sampled signal during rendering is indicated by the red curve. After discrete sampling and filtering, high-frequency details—represented by the red Gaussian and brown Gaussian—are observable only in the zoomed-in view but have negligible effects on the normal view, where high-frequency details are filtered out presented in the right bottom of (b).

## 4.1. Upscaling with Diffusion Models

With the rapid advancement of diffusion models, their sampling procedures and design spaces have been extensively exploited in practical scenarios [23, 46]. Our theoretical framework is primarily based on Latent Diffusion Models (LDMs) [51]. Upscale diffusion models first encode an image into a latent space, and then modify and upsample the latent variables through the reverse diffusion process. Given the original pixel  $p_o \in \mathbb{R}^{1 \times 1 \times 3}$  and a zoomed-in tile  $\mathbf{P} \in \mathbb{R}^{a \times a \times 3}$ ,  $\mathbf{P}$  consists of a set of pixels with high-frequency details generated by upscaling diffusion models using an upscaling factor a. Based on Eq. 2, each pixel  $p^{ij}$  in  $\mathbf{P}$  can be represented:

$$p^{ij} = \sum_{t \in T} \epsilon_{\theta} \left( \sqrt{\bar{\alpha}_t} \, \mathbf{x}_t(p_o, \sqrt{1 - \bar{\alpha}_t} \, \boldsymbol{\epsilon}) \right). \tag{3}$$

We present the equation in this form for simplicity, as upsampling can be seamlessly integrated into the model's forward process.

High-frequency details in  ${\bf P}$  originate from the random noise  $\epsilon$  introduced during each reverse diffusion process [51]. While these details enhance the visual quality of the image, it is essential to ensure they perfectly align with the original lower-frequency ground truth  $p_o$ , as random noise can introduce significant deviations from the original content. Nevertheless, our thorough investigations reveal that the design space for high-frequency details, potentially generated by upscale diffusion models, is quite flexible. This flexibility stems from two primary factors: first,

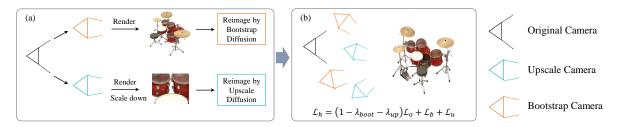


Figure 3. **Visualization of our pipeline.** (a) Starting from an original camera, we first narrow its field of view to construct corresponding zoomed-in cameras and bootstrapping cameras. We render images from these cameras (at a reduced scale for upscaling). These new renderings are then fed into diffusion models for regeneration. (b) The regenerated images are used as pseudo ground truth to contribute to our hybrid loss function.

the rounding function applied after weighted average interpolation; and second, the application of a low-pass filter during the discrete signal reconstruction, particularly in the rendering process of 3D-GS.

## 4.2. Interpolation Flexibility

To recover  $p_o$  from **P** via interpolation, we perform weighted sampling followed by rounding the result to an integer within the range [0, 255]:

$$p_o = Round(\sum_{i}^{a} \sum_{j}^{a} \mathcal{W}(d^{ij})p^{ij}), \tag{4}$$

where 
$$d^{ij} = \sqrt{\left(i - \frac{1}{2}a\right)^2 + \left(j - \frac{1}{2}a\right)^2}$$
. (5)

Here,  $Round(\cdot)$  denotes the rounding function that maps values to their nearest integers within the specified range and  $\mathcal{W}(\cdot)$  is a function to calculate the weight of pixels during interpolation. This rounding function introduces basic flexibility into the design space. If the absolute variation of the pixels in  $\mathbf{P}$  compared to the pixel  $p_o$  does not exceed a threshold  $\tau_v$ , such that the absolute variation after interpolation will not exceed 0.5, and the final result will remain unchanged.

Since these variations are generally introduced by the noise  $\epsilon$  in Eq. 3 during each reverse diffusion step, we assume that given a noise threshold  $\tau_n > 0$ , then for each  $p^{ij}$  in  $\mathbf{P}$ , it holds that  $|p_o - p^{ij}| \le \tau_v$ . This implies:

$$\left| p_o - \sum_{t \in T} \epsilon_{\theta} \left( \sqrt{\bar{\alpha}_t} \, \mathbf{x}_t(p_o, \, \pm \sqrt{1 - \bar{\alpha}_t} \, \tau_n) \right) \right| \le \tau_v. \quad (6)$$

Thus, the problem is reduced to effectively regulating the normal noise  $\epsilon \sim \mathcal{N}(0, I)$  during the diffusion process.

We now focus on controlling the expectation of a set of independent normal distributions. Utilizing Chebyshev's inequality, we can express it in the following form: **Theorem 1** Mathematically, let  $X_1, X_2, ..., X_n$  be i.i.d. random variables with expected value  $\mu = \mathbb{E}[X_i]$  and variance  $\sigma^2 = \text{Var}(X_i) < \infty$ , then, for any  $\varepsilon > 0$ :

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}X_{i}-\mu\right|\geq\varepsilon\right)\leq\frac{\sigma^{2}}{n\varepsilon^{2}}.$$
 (7)

As the sample size n increases, the right-hand side  $\frac{\sigma^2}{n\varepsilon^2}$  approaches zero, indicating that the sample mean converges in probability to the expected value  $\mu$ . And according to **Statistical Principles**:

**Condition 1** When the probability of an event occurring is below a certain threshold, it can be considered negligible for practical purposes in small samples.

Combining these theories, it follows that for a given probability threshold  $\tau_p$ , there exists a sample size  $n_s$  satisfying:

$$\mathbb{P}\left(\left|\frac{1}{n_s}\sum_{i=1}^n \epsilon\right| \ge \tau_n\right) \le \tau_p,\tag{8}$$

such that  $\left|\frac{1}{n_s}\sum_{i=1}^n\epsilon\right|\geq \tau_n$  is highly unlikely to occur during our sampling process, which we could consider to meet the requirements of the inequality in Eq. 6 based on Condition 1.

In practice, we could sacrifice the ability and versatility of diffusion models but directly fix  $\epsilon$  with a small value less than  $\tau_n$ . However, based on our reasoning, we find that increasing the posterior sampling number could also achieve similar results because:

$$\left| \mathbb{E} \left[ \sum_{t \in T} \boldsymbol{\epsilon}_{\theta} \left( \sqrt{\bar{\alpha}_{t}} \, \mathbf{x}_{t}(p_{o}, \, \frac{1}{n_{s}} \sum_{i=1}^{n_{s}} \sqrt{1 - \bar{\alpha}_{t}} \, \boldsymbol{\epsilon}_{i}) \right) \right] \right| \\
\geq \left| \frac{1}{n_{s}} \sum_{i}^{n_{s}} \mathbb{E} \left[ \sum_{t \in T} \boldsymbol{\epsilon}_{\theta} \left( \sqrt{\bar{\alpha}_{t}} \, \mathbf{x}_{t}(p_{o}, \, \sqrt{1 - \bar{\alpha}_{t}} \, \boldsymbol{\epsilon}_{i}) \right) \right] \right|. \tag{9}$$

Therefore, similar to the approach used in Bootstrap-GS, increasing the number of diffusion samples can significantly

reduce misalignment between generated fine-grained details and the original low-frequency ground truth. For a complete derivation and additional details, please refer to our **Supplementary Material**.

## 4.3. Filter Application in 3D-GS

On the other hand, combining the 3D-GS rendering process, to render  $p_o$  from the original viewpoint involves all high-frequency Gaussians visible in the zoomed-in tile  ${\bf P}$ . This inclusion will exceed the highest frequency permissible by the discrete sampling threshold described in Sec. 2. Therefore, a frequency filter  $Fil(\cdot)$  is applied to the Gaussians visible in  ${\bf P}$ . The pixel  $p_o$  rendered from the original view can then be approximated from the interpolation perspective as:

$$p_o = Round(\sum_{i}^{a} \sum_{j}^{a} \mathcal{W}(d^{ij})Fil(p^{ij})).$$
 (10)

In 3D-GS [15, 25, 80], this filter operates on small Gaussians that remain too diminutive to fill an entire pixel, even after extension. During the densification process of 3D-GS, high-frequency details are typically represented by smaller Gaussians. The filter Fil not only acts as a barrier for very small Gaussians but also smooths the distribution of color density during rendering, as illustrated in Figure 2.

By integrating analyses from both the rendering and interpolation perspectives, we conclude that extremely small Gaussians aggregated within a single pixel have a negligible impact on rendering from relatively distant viewpoints. This negligible influence permits us to edit these Gaussians freely without affecting the overall image quality. Conversely, from an interpolation standpoint, the visible zoomed-in high-frequency Gaussians in distant views can still be stably and effectively modified while maintaining alignment with the ground truth of the original views by upscale diffusion models.

## 4.4. Practical Implementation

**Loss Design:** For each training camera  $c_t$  and its corresponding ground-truth image  $\mathbf{I}_t$ , we adjust the translation scale of  $c_t$  to obtain zoom-in cameras  $\mathbf{c}_z$  using a set of scaling factors  $\mathbf{a}$ . We then upscale  $\mathbf{I}_t$  using diffusion models with these scaling factors to generate zoom-in pseudo-ground-truth images  $\mathbf{I}_z$ . Our upscaling loss  $\mathcal{L}_u$  is defined as the average  $\mathcal{L}_1$  loss between the rendered images  $\mathbf{I}_r$  under the zoom-in cameras  $\mathbf{c}_z$  and their corresponding bootstrapped upscaled images  $\mathbf{I}_z$ , where  $\mathcal{L}_u = \frac{\lambda_{\mathrm{up}}}{M} \sum_i^M \|\mathbf{I}_i^r - \mathbf{I}_z^i\|$ . Here M is the total number of zoom-in scales and  $\lambda_{\mathrm{up}}$  is a loss scaling factor. Finally, we employ a hybrid loss  $\mathcal{L}_h$  that combines the original 3D-GS loss  $\mathcal{L}_o$ , the bootstrapping loss  $\mathcal{L}_b$  introduced in Sec. 3.1 (if used), and the upscaling loss  $\mathcal{L}_u$ :  $\mathcal{L}_h = (1 - \lambda_{\mathrm{boot}} - \lambda_{\mathrm{up}})\mathcal{L}_o + \mathcal{L}_b + \mathcal{L}_u$ .

Refinement of Bootstrapping Pipeline: Despite its success, Bootstrap-GS has an unstable training process because of the variability of bootstrapped views. Our extensive experiments show that most fluctuations are not due to multi-view inconsistencies from bootstrapping but rather from the durability of bootstrapped views in early training stages. The rapid completion of 3D scenes means bootstrapped views before several hundred iterations are often largely misaligned with current views from a whole perspective. Additionally, some artifacts cannot be mitigated by merely expanding sampling views. The original 3D-GS generally lacks structural awareness due to its straightforward optimization. Even if artifacts in unseen views are modified, Bootstrap-GS may only adjust colors to appear more "natural," but the underlying Gaussians—which ideally should not exist—remain. As a result, we not only limit the training duration of each bootstrapped view but also use the LOD structure proposed in Octree-GS [50]. For more details and experimental support, please refer to the Supplementary Material.

Cropped Upscaling and Bootstrapping: Upscaling using diffusion models is a time-consuming process. To ensure proper alignment between the generated details and the original content, using extremely small diffusion steps is not acceptable. On top of that, in Bootstrap-GS, bootstrapping time is a significant concern, which may even triple the training time. Combining both upscaling and bootstrapping would render the training time prohibitive. Furthermore, while the color optimization in the original 3D-GS with spherical harmonics is relatively insensitive to bootstrapping alterations as they could be simply converted back independently, MLP-based color prediction methods like Scaffold-GS [36] are highly susceptible to these changes. Altering the whole image, even with a small modification strength via diffusion models, can lead to a complete collapse in the training of the color MLP as it may contradict the previous training results. To address these issues, we have designed a cropping method. We find that reducing the input image size and the camera's field of view not only increases training speed but also improves the final performance for bootstrapping and upscaling. The whole pipeline is displayed in Figure 3. For a detailed analysis, please refer to our Supplementary Material.

## 5. Experiments

## 5.1. Experimental Setup

**Dataset and Metrics.** We conducted comprehensive evaluations on 29 scenes from public datasets, encompassing a diverse range of environments—from complex indoor settings and multi-scale scenarios to large urban landscapes. Specifically, our evaluation includes 9 scenes from Mip-NeRF360 [3], 2 scenes from Tanks & Temples [26], 2

Table 1. Quantitative comparison on real-world datasets [3, 18, 26]. In our Stage 1 experiments, where we utilize the upscaling diffusion model, we observe not only an absence of performance degradation but also slight improvements in the evaluation metrics. In Stage 2, by combining upscaling with bootstrapping, we significantly enhance the overall performance in each scene while maintaining the increment of Gaussian counts in a negligible number. We have highlighted the **best** and <u>second-best</u> results in each category.

Dataset		Mip	-NeRF3	60		Tank	s&Temp	les	Deep Blending				
Method Metrics	PSNR↑	SSIM↑	LPIPS↓	#GS(k)/Mem	PSNR↑	SSIM↑	LPIPS↓	#GS(k)/Mem	PSNR↑	SSIM↑	LPIPS↓	#GS(k)/Mem	
Mip-NeRF360 [3]	27.69	0.792	0.237	-	23.14	0.841	0.183	-	29.40	0.901	0.245	-	
3D-GS [25]	27.54	0.815	0.216	937/786.7M	23.91	0.852	0.172	765/430.1M	29.46	0.903	0.242	398/705.6M	
Mip-Splatting [80]	27.61	0.816	0.215	1013/838.4M	23.96	0.856	0.171	832/500.4M	29.56	0.901	0.243	410/736.8M	
Scaffold-GS [36]	27.90	0.815	0.220	666/197.5M	24.48	0.864	0.156	626/167.5M	30.28	0.909	0.239	207/125.5M	
Boot-3D-GS [15]	28.32	0.823	0.209	951/798.5M	24.85	0.863	0.163	788/448.3M	31.43	0.914	0.231	412/730.4M	
Octree-GS [50]	28.05	0.819	0.214	657/ <b>139.6M</b>	24.68	0.866	0.153	443/ <b>88.5M</b>	30.49	0.912	0.241	112/ <b>71.7M</b>	
Octree-UP	28.14	0.819	0.213	664/ <u>141.2M</u>	24.78	0.868	0.152	463/ <u>92.5M</u>	30.56	0.914	0.239	114/ <u>73.1M</u>	
Octree-UB	28.53	0.825	0.210	674/143.3M	25.15	0.871	0.149	474/94.7M	31.23	0.919	0.237	112/116.9M	

scenes from DeepBlending [18], 8 scenes from BungeeN-eRF [69], and 8 objects from NeRF-Synthesis [39].

Our primary focus is on comprehensively improving scene reconstruction without altering the structure of our frameworks. Therefore, we report only standard visual quality metrics such as PSNR, SSIM [67], LPIPS [82], and the number of Gaussian primitives. For other relevant metrics like FPS, the variations compared to the original models remain minimal. The quantitative metrics averaged over all scenes and test frames are presented in the main paper, while detailed results for each scene are provided in the **Supplementary Material**.

Baseline and Implementation. We compare our method against the original 3D-GS [25], Mip-Splatting [80], Bootstrap-GS [15], the original Scaffold-GS [36], and Octree-GS [50]. We also report the results of MipN-eRF360 [3] for rendering quality comparisons. To ensure fair comparisons with the original results, we kept all modeling parameters unchanged and only incorporated bootstrapping and upscaling techniques. Additionally, as Octree-GS [50] has been implemented across various frameworks, including 3D-GS and Scaffold-GS [36], we focus primarily on its most successful version, the Scaffold-GS implementation. In the following sections, all references to Octree-GS default to its implementation on Scaffold-GS.

Due to the distinct characteristics of each dataset, we apply task-specific training schedules. For foundational outdoor and indoor datasets, Mip-NeRF360, Tanks& Temples, and DeepBlending, we conduct two-stage training per scene using the Octree-GS framework. In the first stage, we incorporate upscaling with a factor of a=2, denoted as **Octree-UP**, leveraging the open-source diffusion model **SD-X2-Latent-Upscaler** [51] to generate zoom-in details, which are reintegrated into training. In the second stage, in addi-

tion to upscaling, we integrate bootstrapping using **Stable Diffusion SDXL-Turbo** along with our refined bootstrapping pipeline, denoted as **Octree-UB**. During this stage, we fine-tune **SDXL-Turbo** based on [15] for each scene while keeping the scaling factor unchanged. Given that the Octree-GS version we use here, which relies on MLP-based color prediction, is sensitive to bootstrapping disturbances as described in Section 4.4, we also train an additional version using Octree-3D-GS as our framework. This framework is an implementation of Octree-GS based on the original 3D-GS structure.

For the multi-scale dataset BungeeNeRF, we apply only upscaling diffusion models on Octree-GS, conducting twostage training with different scaling factors. Specifically, in the first stage, we use a scaling factor of a = 2, denoted as Octree-UPx2, and in the second stage, we apply scaling factors  $a = \{2, 4\}$ , denoted as **Octree-UPx4**. For NeRF-Synthesis, we apply upscaling diffusion models with scaling factors  $\mathbf{a} = \{4, 8\}$  on the original 3D-GS (UP-**3DGS**) and Mip-Splatting (UP-Mip) frameworks, as the LOD structure is ineffective in this context, and MLP-based color prediction tends to produce lower-frequency details in highly zoomed-in views. The upscaling diffusion models used are SD-X2-Latent-Upscaler for lower scaling factors and SD-X4-Upscaler for higher scaling factors across each dataset. For further implementation details and explanations, please refer to our **Supplementary Material**.

#### **5.2. Results Analysis**

In both outdoor and indoor scene results, as shown in Table 1, Figures 4 and 6 (left), our first-stage experiment, **Octree-UP**, effectively preserves original content while enhancing fine details that are otherwise obscured in the ground truth. In the second stage, **Octree-UB**, our refined bootstrapping pipeline builds on **Octree-UP** to achieve substantial metric improvements with moderate added training time (see Sec. 5.3). Additionally, training within the **Octree-3D**-

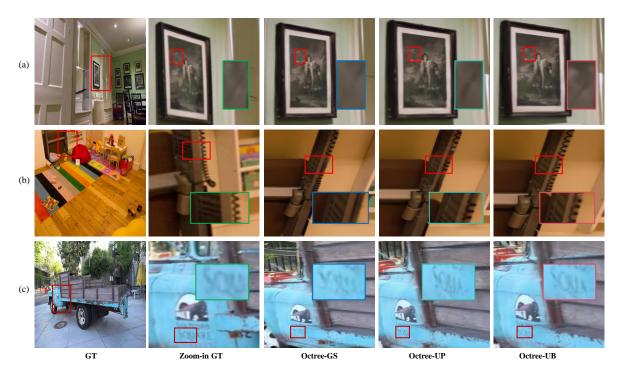


Figure 4. **Main comparisons.** (a) For extremely small details present in the ground truth images, our method effectively completes and reconstructs these details. (b) In scenarios involving partially occluded views within the training datasets, our random upscale sampling technique enables the generation of fine-grained details that align with the ground truth. (c) For vague or indistinct details even in the ground truth, our upscaling diffusion models are capable of denoising the ground truth and generating high-frequency details. **Note**: All our renderings are produced from zoomed-in views, for which there is no directly aligned ground truth available.

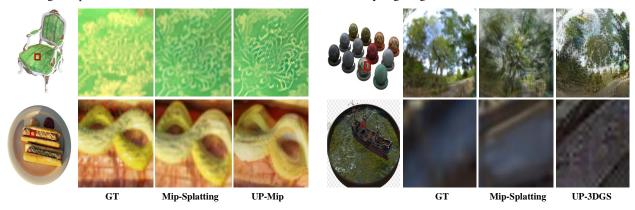


Figure 5. **Object rendering comparisons.** Our method enables the generation of fine-grained, flexible, and scene-aligned details using upscaling diffusion models on zoomed-in scales without impacting the integrity of the original rendering on normal scales.

**GS** framework produces further gains, significantly outperforming the original Bootstrap-GS, as detailed in our **Supplementary Material**.

For the multi-scale datasets shown in Table 2, we observe that in most scenarios, fine-grained details generated by upscaling diffusion models are largely imperceptible in the test data, while only visible in zoomed-in views as shown in Figure 1. Consequently, modifications mainly impact the overall structure in training, resulting in only minor metric improvements. However, as scale increases, the benefits of

our upscaling method become clearer, with structural adjustments to the scenes contributing more substantially to performance gains.

Finally, for object training, each framework offers unique advantages, as shown in Figure 5. Under **UP-Mip**, with the application of a 3D smoothing filter and a 2D Mip filter, high-frequency details appear smoother. In contrast, **UP-3DGS** produces sharper, though occasionally more variable, details. Both methods demonstrate the ability to generate high-frequency details out of training data

Table 2. Quantitative comparison on the BungeeNeRF [69] dataset. We present evaluation metrics for each scale and their averages across all four scales. Scale-1 corresponds to the closest viewpoints, while Scale-4 encompasses the entire landscape. Our method shows increasing effectiveness as the scale increases, all while maintaining a minimal increase in the number of Gaussians.

Dataset		Bungee	NeRF (A	verage)	Scale-1		Scale-2		Scale-3		Scale-4	
Method   Metrics	PSNR↑	SSIM↑	$LPIPS\!\!\downarrow$	#GS(k)/Mem	PSNR↑	#GS(k)	PSNR↑	#GS(k)	PSNR↑	#GS(k)	PSNR↑	#GS(k)
3D-GS [25]	27.79	0.917	0.093	2686/1792.3M	30.00	522	28.97	1272	26.19	4407	24.20	5821
Mip-Splatting [80]	28.14	0.918	0.094	2502/1610.2M	29.79	503	29.37	1231	<u>26.74</u>	4075	24.44	5298
Scaffold-GS [36]	28.16	0.917	0.095	1652/319.2M	30.48	303	29.18	768	26.56	2708	24.95	3876
Octree-GS [50]	28.39	0.923	0.088	1474/ <b>296.7M</b>	31.11	486	29.59	1010	26.51	2206	25.07	2167
Octree-UPx2	28.43	0.923	0.088	1521/ <u>306.2M</u>	31.14	490	29.63	1022	26.83	2234	25.28	2191
Octree-UPx4	28.48	0.924	0.087	1568/315.6M	31.08	493	29.72	1031	26.75	2249	25.41	2213

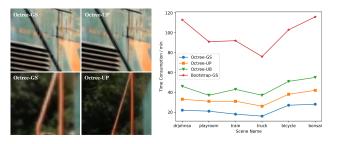


Figure 6. Left: Visualization of zoomed-in  $\times 4$  views between original Octree-GS and our Octree-UP in **Train** [26]. Our renderings exhibit less dilation and sharper details. Right: Time comparisons among different methods. Our Octree-UP and Octree-UB achieve significant progress on performance and largely reduce the time consumption compared with Bootstrap-GS.

compared with blurry ground truth, significantly enriching 3D reconstruction results. Further analyses are in our **Supplementary Material**.

### 5.3. Further Details

Time Consumption: Our training process is significantly faster than the original bootstrapping pipeline, as shown in Figure 6 (right). With the upscale diffusion model, Octree-UP requires only 10 additional minutes for outdoor and indoor scenes. Even with bootstrapping integration, our training time of Octree-UB averages under one hour per scene. For the multi-scale training in BungeeNeRF datasets [69], we only upscale part of the training data, which further accelerates the training process. Additionally, We also trained the original model for the same length of time for fair comparisons. The results indicate that extended training contributes minimal improvements in metrics and does not further alleviate zoomed-in artifacts, but even worsens. For more details please refer to our Supplementary Material.

**Flexibility of Zoom-in Details:** Our experiments reveal that in most scenes small artifacts like "partial breakage" and minor distortion begin to appear when zooming in 2×

on the original camera views. When zooming in  $4\times$ , some regions exhibit complete breakdown. This suggests that the more a reconstructed scene collapses in zoomed-in views, the more flexibility we have to use upscale diffusion models. In these views, conditional upscaling generation can even be used to precisely design the desired details. For instance, in Figure 5, our generation may slightly deviate from the original content, yet the evaluation metrics in standard views remain comparable to original frameworks.

Control of Zoom-in Details: To control the flexibility of zoom-in details, we could finetune the upscale diffusion models to achieve more aligned generation. Additionally, we could apply a gradual zoom-in scaling approach, for instance, using a set of scales such as  $\{2,4,6,8\}$  rather than our  $\{4,8\}$  for object reconstruction, to progressively refine details at each zoom level. The effectiveness of this approach could be validated through the application of BungeeNeRF in Figure 1, where generated details closely align with the original content. However, in our paper, we opted not to implement this gradual approach because we aimed to demonstrate the flexibility of the design space. Instead, we present it as a solution for achieving higher fidelity in practical implementations.

#### 6. Conclusion

We introduce a new training pipeline for 3D-GS that integrates both Bootstrap-GS and upscaling diffusion models. Our approach includes a comprehensive theoretical analysis and has demonstrated remarkable practical success. It significantly reduces zoom-in artifacts commonly seen in diverse 3D-GS frameworks and leverages a flexible design space for detailed zoom-in elements, enhancing the expressive quality of 3D reconstruction. Additionally, our refined bootstrapping and upscaling pipelines result in a much faster training process compared to the original Bootstrap-GS. Our method achieves state-of-the-art results with improved generalization to out-of-distribution camera poses, all while adding minimal extra training time.

#### References

- [1] Haotian Bai, Yiqi Lin, Yize Chen, and Lin Wang. Dynamic plenoctree for adaptive sampling refinement in explicit nerf. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8785–8795, 2023.
- [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 *Proceedings of the IEEE/CVF Inter*national Conference on Computer Vision, pages 5855–5864, 2021.
- [3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 5, 6, 16, 20, 21
- [4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pages 19640–19648, 2023.
- [5] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023.
- [6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In European Conference on Computer Vision, pages 333–350. Springer, 2022. 2
- [7] Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3d using gaussian splatting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 21401–21412, 2024.
- [8] Bardienus P. Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Jenny Seidenschwarz, Mike Zheng Shou, Deva Ramanan, Shuran Song, Stan Birchfield, Bowen Wen, and Jeffrey Ichnowski. Deformgs: Scene flow in highly deformable scenes for deformable object manipulation, 2024. 1
- [9] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps, 2024. 1
- [10] Hao Fang, Florent Lafarge, and Mathieu Desbrun. Planar shape detection at structural scales. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2965–2973, 2018.
- [11] Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, and Yin Yang. Gaussian splashing: Unified particles for versatile motion synthesis and rendering, 2024.
- [12] Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, et al. Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. *arXiv preprint arXiv:2401.15318*, 2024.

- [13] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.
- [14] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.
- [15] Yifei Gao, Jie Ou, Lei Wang, and Jun Cheng. Bootstrap 3d reconstructed scenes from 3d gaussian splatting, 2024. 1, 3, 5, 6, 13, 14, 15, 16, 17, 18, 20, 21
- [16] Zhiyang Guo, Wengang Zhou, Li Li, Min Wang, and Houqiang Li. Motion-aware 3d gaussian splatting for efficient dynamic scene reconstruction, 2024. 1
- [17] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. Ges: Generalized exponential splatting for efficient radiance field rendering. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 19812–19822, 2024. 1
- [18] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (ToG), 37(6):1–15, 2018. 6, 17, 20
- [19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020. 2, 3, 13
- [20] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4220–4230, 2024.
- [21] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, and Chenfanfu Jiang. Vr-gs: A physical dynamicsaware interactive gaussian splatting system in virtual reality. In ACM SIGGRAPH 2024, 2024. 1
- [22] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *International Journal of Computer Vision (IJCV)*, 129(2):517–547, 2021.
- [23] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. Advances in neural information processing systems, 35:26565–26577, 2022. 3
- [24] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track map 3d gaussians for dense rgb-d slam. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 21357–21366, 2024.
- [25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics, 42 (4), 2023. 1, 3, 5, 6, 8, 14, 16, 20, 21

- [26] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG), 36 (4):1–13, 2017. 5, 6, 8, 15, 20
- [27] Samuli Laine and Tero Karras. Efficient sparse voxel octrees. IEEE Transactions on Visualization and Computer Graphics, 17(8):1048–1059, 2011.
- [28] Jiameng Li, Yue Shi, Jiezhang Cao, Bingbing Ni, Wenjun Zhang, Kai Zhang, and Luc Van Gool. Mipmap-gs: Let gaussians deform with scale-specific mipmap for anti-aliasing rendering. In *International Conference on 3D Vision*, 2025. 1, 2, 21
- [29] Yixuan Li, Lihan Jiang, Linning Xu, Yuanbo Xiangli, Zhenzhi Wang, Dahua Lin, and Bo Dai. Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3205–3215, 2023.
- [30] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8456–8465, 2023.
- [31] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 6517–6526, 2024.
- [32] Liqiang Lin, Yilin Liu, Yue Hu, Xingguang Yan, Ke Xie, and Hui Huang. Capturing, reconstructing, and simulating: the urbanscene3d dataset. In *European Conference on Computer Vision (ECCV)*, pages 93–109. Springer, 2022.
- [33] Kunhao Liu, Fangneng Zhan, Muyu Xu, Christian Theobalt, Ling Shao, and Shijian Lu. Stylegaussian: Instant 3d style transfer with gaussian splatting. In SIGGRAPH Asia, 2024.
- [34] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. Advances in Neural Information Processing Systems, 33:15651–15663, 2020.
- [35] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. arXiv preprint arXiv:2309.03453, 2023. 2
- [36] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 20654–20664, 2024. 1, 5, 6, 8, 14, 18, 20,
- [37] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In 2024 International Conference on 3D Vision (3DV), pages 800–809, 2024.
- [38] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: adaptive coordinate networks for neural scene representation. ACM Trans. Graph., 40(4), 2021.

- [39] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7210–7219, 2021. 6
- [40] 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. *Communications of the ACM*, 65(1):99–106, 2021.
- [41] 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.
- [42] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compgs: Smaller and faster gaussian splatting with vector quantization. In European Conference on Computer Vision (ECCV), 2024. 1
- [43] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, pages 8162–8171. PMLR, 2021. 2, 13
- [44] Harry Nyquist. Certain topics in telegraph transmission theory. Transactions of the American Institute of Electrical Engineers, 47(2):617–644, 1928. 2
- [45] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 5865–5874, 2021.
- [46] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *International Con*ference on Learning Representations (ICLR), 2024. 3
- [47] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv* preprint arXiv:2209.14988, 2022. 3
- [48] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 20299–20309, 2024.
- [49] Christian Reiser, Stephan Garbin, Pratul Srinivasan, Dor Verbin, Richard Szeliski, Ben Mildenhall, Jonathan Barron, Peter Hedman, and Andreas Geiger. Binary opacity grids: Capturing fine geometric detail for mesh-based view synthesis. ACM Trans. Graph., 43(4), 2024.
- [50] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-GS: Towards consistent real-time rendering with lod-structured 3d gaussians, 2024. 1, 5, 6, 8, 14, 15, 16, 17, 19, 20, 21
- [51] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of* the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10684–10695, 2022. 3, 6

- [52] Steven M Rubin and Turner Whitted. A 3-dimensional representation for fast rendering of complex scenes. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 110–116, 1980.
- [53] Shunsuke Saito, Gabriel Schwartz, Tomas Simon, Junxuan Li, and Giljoo Nam. Relightable gaussian codec avatars. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 130–141, 2024. 1
- [54] Johannes L Schonberger and Jan-Michael Frahm. Structurefrom-motion revisited. In *Proceedings of the IEEE con*ference on computer vision and pattern recognition, pages 4104–4113, 2016.
- [55] Claude Elwood Shannon. Communication in the presence of noise. Proceedings of the IRE, 37(1):10–21, 1949.
- [56] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015. 2, 13
- [57] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021. 3, 13
- [58] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. 2, 13
- [59] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5459– 5469, 2022.
- [60] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8248–8258, 2022.
- [61] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *International Conference on Learning Representations (ICLR)*, 2023. 3
- [62] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *Computer Vision – ECCV 2024*, pages 1–18, 2025.
- [63] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of largescale nerfs for virtual fly-throughs. In *Proceedings of* the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12922–12931, 2022.
- [64] Haithem Turki, Michael Zollhöfer, Christian Richardt, and Deva Ramanan. Pynerf: Pyramidal neural radiance fields. Advances in Neural Information Processing Systems, 36: 37670–37681, 2024.
- [65] Yannick Verdie, Florent Lafarge, and Pierre Alliez. LOD Generation for Urban Scenes. ACM Trans. on Graphics, 34 (3), 2015.

- [66] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12619–12629, 2023. 3
- [67] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6
- [68] 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 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 20310–20320, 2024. 1
- [69] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In European Conference on Computer Vision (ECCV), pages 106–122, 2022. 6, 8, 15, 19, 21
- [70] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Bo Dai, and Dahua Lin. Assetfield: Assets mining and reconfiguration in ground feature plane representation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 3251–3261, 2023.
- [71] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physicsintegrated 3d gaussians for generative dynamics. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4389–4398, 2024. 1
- [72] Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, et al. Vr-nerf: Highfidelity virtualized walkable spaces. In SIGGRAPH Asia 2023, pages 1–12, 2023.
- [73] Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8296–8306, 2023.
- [74] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022.
- [75] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians for modeling dynamic urban scenes. In *European Conference on Computer Vision* (ECCV), 2024. 1
- [76] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 20331–20341, 2024.
- [77] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes.

- In The European Conference on Computer Vision (ECCV), 2024. 1
- [78] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752– 5761, 2021.
- [79] Mulin Yu and Florent Lafarge. Finding Good Configurations of Planar Primitives in Unorganized Point Clouds. In *Proc. of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6357–6366, 2022.
- [80] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 19447–19456, 2024. 1, 2, 5, 6, 8, 20, 21
- [81] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. arXiv preprint arXiv:2312.10070, 2023.
- [82] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 6
- [83] Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. Gpsgaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 19680–19690, 2024. 1
- [84] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 21634–21643, 2024. 1
- [85] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3d gaussian avatars. arXiv preprint arXiv:2311.08581, 2023. 1
- [86] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization*, 2001. VIS '01., pages 29–538, 2001. 13

## 7. Supplementary Material

**Overview.** This supplementary is structured as follows: (1) The first section elaborates on our further analyses and implementation details, (2) and additional experimental results are also presented.

**Erratum.** We have identified an error in Table 1, which presents the main outdoor and indoor comparison. Specifically, the values for #GS(k)/Mem associated with Deep Blending are incorrect. The correct values should be 117/75.6M.

## 7.1. 3D Gaussian Splatting

The geometry of each scaled 3D Gaussian starts with a set of points derived from Structure-from-Motion (SfM), each point is designated as the position (mean)  $\mu$  of a 3D Gaussian:  $G(x) = e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$ . To constrain  $\Sigma$  to the space of valid covariance matrices, a semi-definite parameterization is used. Then, the covariance matrix  $\Sigma$  is constructed utilizing a scaling matrix  $\Sigma$  and a rotation matrix  $\Sigma$ , ensuring that it remains positive semi-definite:  $\Sigma = RSS^TR^T$ .

To render an image for a given viewpoint efficiently, 3D-GS uses tile-based rasterization. This process involves initially converting the 3D Gaussians G(x) into 2D Gaussians G'(x) on the image plane as a result of the projection process described in [86]. Subsequently, a specially designed tile-based rasterizer sorts these 2D Gaussians and applies  $\alpha$ -blending:

$$C(x') = \sum_{i \in N} c_i \sigma_i \prod_{j=1}^{i-1} (1 - \sigma_j), \quad \sigma_i = \alpha_i G'_i(x'), \quad (11)$$

where x' is the queried pixel position and N denotes the number of sorted 2D Gaussians associated with the queried pixel.

#### 7.2. Theoretical Reasoning Completion

In this section, we aim to complete the logical chain outlined in Sec. 4.2. We begin by examining the left-hand side of the inequality in Eq. 9, which is given by:

$$\left| \mathbb{E} \left[ \sum_{t \in T} \epsilon_{\theta} \left( \sqrt{\bar{\alpha}_t} \, \mathbf{x}_t(p_o, \, \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{1 - \bar{\alpha}_t} \, \epsilon_i) \right) \right] \right|. \quad (12)$$

As discussed in Sec. 4.2, this term can be interpreted as a a global sample with an averaged collection of  $n_s$  noise terms for each diffusion time step. Each averaged collection of noise terms attains the maximum expected value satisfying the inequality in Eq. 6 based on Theorem 1. Conversely, the right-hand side of the inequality in Eq. 9 is expressed as:

$$\left| \frac{1}{n_s} \sum_{i}^{n_s} \mathbb{E} \left[ \sum_{t \in T} \epsilon_{\theta} \left( \sqrt{\bar{\alpha}_t} \, \mathbf{x}_t(p_o, \sqrt{1 - \bar{\alpha}_t} \, \epsilon_i) \right) \right] \right|. \quad (13)$$

This expression can be viewed as averaged  $n_s$  global samples, each with a single noise term for each diffusion time step.

To simplify the analysis, we omit the term  $p_o$  in the function  $\mathbf{x}_t$  since it remains constant in both Eq. 12 and Eq. 13, and there is no significant dependence between  $p_o$  and  $\epsilon$ . Consequently,  $\mathbf{x}_t$  becomes a function solely of  $\epsilon$ . And because  $\bar{\alpha}_t$  is a pre-defined constant, we can merge  $\epsilon_{\theta}\left(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_t(\sqrt{1-\bar{\alpha}_t}\,\epsilon_i)\right)$  into a simple function  $\mathbf{f}_t(\epsilon)$ . By applying **Jensen's Inequality**, we reduce the problem to examining the convexity or concavity of the function  $\mathbf{f}_t(\epsilon)$  for each diffusion time step t.

However, since  $\epsilon_{\theta}$  is a neural network comprising convolutions and nonlinear activation functions, it does not exhibit clear convexity or concavity properties. Therefore, we shift our focus to the behavior of its latent space to approximate the function's response. Specifically, we investigate whether large variations in the sampled noise  $\epsilon$  lead to significant fluctuations in the output.

To address this, we consider the properties of diffusion models as defined in [19, 43, 57]. During the reverse diffusion process, the term  $\sqrt{1-\bar{\alpha}_t}$  diminishes progressively to zero at an exponential rate due to the cumulative product. This allows us to neglect time steps that contribute minimal variations and only focus on the initial reverse diffusion steps, which have a significant impact on the model's output.

Furthermore, based on the foundational theory of nonequilibrium thermodynamics in diffusion models [56], a well-trained diffusion model is capable of recovering the original image from completely random noise after infinite reverse diffusion steps. This implies that initial noise fluctuations have a negligible effect on the final output. The model inherently possesses robustness, constraining large noise perturbations introduced in earlier time steps within an acceptable range.

Additionally, as demonstrated in DCSN [58], to address distribution challenges in low-dimensional spaces, noise is added to the original data during training to smooth the distribution, yet diffusion models still achieve successful results. This indicates that the latent space of diffusion models is sufficiently expressive such that local noise with significant variations in earlier time steps does not disrupt the expected distribution in the final output.

Therefore, we can reasonably assume that  $\mathbf{f}_t$  behaves as a concave function satisfying the inequality  $\mathbf{f}_t(\mathbb{E}[\epsilon]) \geq \mathbb{E}\left[\mathbf{f}_t(\epsilon)\right]$ . Under this assumption, the inequality in Eq. 9 holds true, completing the logical chain of our argument.

## 7.3. Analysis of Bootstrapping Deficiency

This section aims to provide explanations for Sec. 4.4. As outlined in Bootstrap-GS [15], Gaussians with higher variations in gradient direction after bootstrapping tend to gener-

ate relatively consistent lower-frequency details due to the average bootstrapping loss. This enhancement improves evaluation metrics without causing significant fluctuations during training. On the other hand, the majority of out-of-view Gaussians—those with small sizes and fine-grained details—are generally unseen or occluded during rendering in test views, thanks to the filtering process introduced in Sec. 4.3. These Gaussians are essentially undetectable unless one explores novel-view renderings far beyond the training and testing sets. Consequently, bootstrapping effectively serves its intended purpose without introducing large training fluctuations.

However, the issue of large training fluctuation arises from the mismatch between the bootstrapping frequency and the rate of scene integration during training. Specifically, in the early stages of training, the reconstruction progresses so rapidly that after several hundred iterations, the current views represented by the trained Gaussian model differ significantly from those rendered a few hundred iterations earlier. Bootstrapping these misaligned views further introduces distortion. Consequently, during the early training phase, the bootstrapped views are significantly misaligned with the overall context and contribute minimally to unseen views after subsequent iterations. This misalignment can even lead to the degradation of the entire scene reconstruction.

On the other hand, the Gaussian model proposed in 3D Gaussian Splatting (3D-GS) [25] employs a straightforward optimization process. During reconstruction, when misalignments occur, the Gaussian model disperses Gaussian primitives to fill those regions with visually consistent colors. However, due to the incompleteness of scenes and a deficiency in training sampling, some excessively dispersed Gaussian primitives remain unmodified. Even with the application of bootstrapping, these Gaussians can easily adjust their Spherical Harmonics to compensate for misaligned areas without reducing their opacities. Consequently, Gaussian primitives that ideally should not exist persist throughout the reconstruction.

#### 7.4. Cropped Bootstrapping

For bootstrapping, we proactively narrow the field of view of the novel-view camera to limit the size of the rendered image by a scale factor  $s_{nar}$ . In our experiments, we set  $s_{nar}=0.4$ . As a result, the bootstrapped image becomes  $\frac{4}{25}$  of its original size, which largely reduces the bootstrapping time. Moreover, our experiments indicate that most parts of the bootstrapped images exhibit no strong artifacts. While the color optimization in the original 3D-GS with spherical harmonics is relatively insensitive to bootstrapping alterations as they could be simply converted back independently, MLP-based color prediction methods like Scaffold-GS [36] are highly sensitive to changes in the en-

tire image. Altering the whole image, even with a small modification strength via diffusion models, can lead to a complete collapse in the training of the color MLP as it may contradict the previous training results. Therefore, cropping the bootstrapped image is conducive for stable training.

## 7.5. Cropped Upscaling

While the details generated by diffusion models during upscaling are generally consistent with the original content, the generation process is computationally intensive. Furthermore, zoomed-in views often exhibit noticeable artifacts. If we employ upscaling diffusion models to amplify these artifacts, the results may become unstable. To address this issue, we reduce both the field of view of the zoomed-in camera used for upscaling and scale down the rendering size by a factor of a. For example, with an upscale factor a=2and the narrow factor  $s_{nar} = 0.5$ , we render the image at  $\frac{1}{4}$  of its original height and width, resulting in a rendered image that is  $\frac{1}{16}$  of its original size. Our approach is based on the same principle described in Sec. 4.3: enlarging the pixel size can effectively filter out zoomed-in small Gaussian primitives. Zoom-in artifacts occur only when sizes of Gaussian primitives are large enough to span at least one pixel. By reducing the rendering size, we use a smaller image to encompass more content. Therefore, further decreasing the rendering size is analogous to the effects depicted in Fig. 2(b), where the pixel size is extended. Subsequently, after applying upscaling, we obtain a more faithful render-

#### 7.6. Implementation details

Our experimental setup for each framework model is consistent with that of Octree-GS [50]. All scenes are trained on NVIDIA RTX 4090 GPUs.

#### 7.6.1. Basic Configurations

Following the methodology of Octree-GS [50], we trained all frameworks on basic indoor, outdoor, and multi-scale datasets for 40,000 iterations. In the Octree-Scaffold-GS framework, densification begins at iteration 500, while in the Octree-3D-GS framework, it starts at iteration 1,500. For both frameworks, the densification process concludes at iteration 25,000. For object training, we trained both Mip-Splatting and 3D-GS for 30,000 iterations. In these cases, densification commences at iteration 1,500 and ends at iteration 15,000. All other configurations, such as the learning rate and loss rate schedule, remained unchanged from their original frameworks.

#### 7.6.2. Bootstrapping Configurations

Compared to Bootstrap-GS [15], we increased the frequency of bootstrapping but delayed its implementation in our experiments. Bootstrapping was initiated at iteration 20,000 and concluded at iteration 38,000. Specifically,

Table 3. Results for Tanks&Temples [26] with different training time. Here, we denote Octree-Scaffold-GS as Octree-GS for simplicity. And for iteration counts, each number refers to  $\times 10,000$  iterations.

Dataset		Truck			Train	
Method Metrics	PSNR (↑)	$SSIM \ (\uparrow)$	LPIPS $(\downarrow)$	PSNR (↑)	$SSIM \ (\uparrow)$	LPIPS $(\downarrow)$
Octree-GS-4 [50]	26.24	0.894	0.122	23.11	0.838	0.184
Octree-GS-5	26.21	0.893	0.122	23.07	0.837	0.185
Octree-GS-6	26.22	0.894	0.122	23.08	0.837	0.184
Octree-GS-7	26.22	0.893	0.122	23.07	0.837	0.185
Octree-GS-8	26.24	0.894	0.122	23.12	0.838	0.184
Octree-UB	26.47	0.896	0.120	23.83	0.846	0.179

we performed bootstrapping every 2,000 iterations. During each bootstrapping interval, we conducted 750 iterations. This application ensures bootstrapped views are properly integrated and aligned with the ongoing training of the scene.

For each training camera, we construct 2 of its randomly generated cameras and put them back to training the same as in Bootstrap-GS. For random cameras, we altered both the rotation matrices  $\boldsymbol{R}$  and the translation vectors  $\boldsymbol{t}$  by adding random noise with scaling factors of 0.2 and 0.1, respectively (after which  $\boldsymbol{R}$  was re-normalized to ensure it remained a valid rotation matrix). Then we use a narrowing factor of 0.4 to scale down the field of view of bootstrapped cameras, with its rendered size reduced accordingly. Additionally, with structural improvement, the implementation of upscale diffusion models in Bootstrap-GS [15] becomes no longer unnecessary, so we discard such integration.

The implementation of bootstrapping diffusion models is relatively flexible, and customized diffusion models can also be effective. We utilize the text prompt "sharp, denoise, original content, natural, detailed, 8k" to guide conditional generation, employing a scaling factor of 1.0. Based on prior investigations [15] and our experimental results, we find that as long as the sampling number is sufficiently large, minor adjustments in the configurations of diffusion models offer negligible improvements. Furthermore, the influence of the inference time step is minimal. In our experiments, we employ a linearly decaying guidance strength ranging from 0.06 to 0.01. For each value of the guidance strength, we dynamically adjust the total number of inference steps but perform only a single inference step during training.

#### 7.6.3. Upscaling Configurations

The upscaled cameras are derived directly from the training cameras by adjusting their scale to create zoomed-in views. Specifically, for each training camera, we modify its scale parameter to obtain a corresponding zoomed-in camera. Similar to the bootstrapping process, we add 2 addi-

tional random cameras for each training camera, altering only their rotation parameters randomly while keeping the scaling factor constant across all expanded zoomed-in cameras. Additionally, for multi-scale datasets BungeeNeRF [69], we only apply our upscaling approach on the first 100 training cameras, as these cameras are gradually zoomed out, which further reduces the training time.

We initiate the upscaling process at iteration 22,000 and conclude it at iteration 38,000, using an upscaling interval of 2,000 iterations. Unlike bootstrapping, which can introduce training disturbances, upscaled regeneration remains relatively faithful to the original data. Therefore, during each interval, we incorporate these upscaled cameras for 1,000 iterations of training, leaving the remaining 1000 iterations unchanged. The narrowing factor is set to 0.5, effectively reducing the field of view by half, which is also sufficient to maintain training efficiency.

In configuring the upscaling diffusion model, we fix the inference time step at 15 for all experiments. As demonstrated in Sec. 4.2 and Sec. 4.3, the design space permits flexibility for fine-grained details, so we employ the conditional generation with prompt "sharp, detailed, denoise, 8k". To ensure the alignment of details generated by the upscaling diffusion models, we set the noise generator seed to a small value of 22, and the conditional guidance for upscaling is merely 1.0.

#### 7.6.4. Loss Weights

We assign different values to the bootstrapping loss weight  $\lambda_b$  and the upscaling loss weight  $\lambda_u$  in Sec. 4.4 due to scene alignment issues discussed in Section 4.4. Specifically, we set  $\lambda_b = [0.15, 0.1]$  and  $\lambda_u = [0.1, 0.05]$ . Both of these factors are very small because of the disturbance issue on color prediction MLP. We uniformly divide the training duration of bootstrapping and upscaling in each interval and apply different scales to their respective losses accordingly. Additionally, the upscale loss could be amplified and directly added to the training loss without the scaling factor  $\lambda_u$  in no-color-MLP frameworks, as modifications on zoomed-in

Gaussian primitives are generally invisible in normal views. In the object training **UP-Mip** and **UP-3D-GS**, the upscaling loss is added directly.

## 8. Experiments

## 8.1. Performance on Different Training Time

Based on our experiments and an analysis of the original 3D Gaussian Splatting (3D-GS) results [25], we observe that beyond a certain performance threshold, further increasing the training time does not lead to improved performance. Specifically, the results from the original 3D-GS indicate that training for 70,000 iterations yields similar or even degraded performance compared to training for 30,000 iterations. In our study, we additionally trained the Octree-Scaffold-GS [50] model for 80,000 iterations. Consistent with the observations from the original 3D-GS, the Octree-Scaffold-GS also encounters a performance bottleneck, as shown in Table 3. Note that the time required to train for 55,000 iterations is equivalent to that of our **Octree-UP** while training for 70,000 iterations is comparable in time to our **Octree-UB**.

Moreover, extending the training time does not alleviate the zoomed-in artifacts. We trained Octree-Scaffold-GS on the multi-scale dataset BungeeNeRF for 70,000 iterations, which exceeds the training time required by our Octree-UP-x4 method. Despite the prolonged training, we observed no improvement in performance metrics, and the zoomed-in artifacts persisted without mitigation; in fact, they were even exacerbated, as shown in Figure 7. These results further demonstrate the effectiveness and necessity of our methods.

Through our experiments, we have found that this issue occurs because almost all regions exhibiting strong artifacts are not visible in the training dataset, and any modifications to the appearance of these Gaussian primitives are made during the early stages of training. After the scene reconstruction is relatively completed, these Gaussian primitives become occluded and are no longer visible from any training perspective, causing their features to remain unchanged during further training. However, the color MLP continues to evolve throughout the training process. As the color MLP in Octree-Scaffold-GS undergoes changes, the previous features are unable to generate the same colors with the modified MLP. Consequently, artifacts in these areas become exacerbated through further training.

To further validate the effectiveness of our method, we conducted extended training of the Octree-3D-GS model on the BungeeNeRF dataset. With the increased training time, its artifacts did not worsen; however, they remained unchanged throughout the additional training. This observation is consistent with the findings reported in the original 3D-GS study [25], where prolonged training similarly failed

Table 4. Quantitative comparison of different frameworks on Mip-NeRF360 [3] datasets with our methods.

Method	PSNR(↑)	SSIM(↑)	LPIPS(↓)
3D-GS [25]	27.54	0.815	0.216
Octree-3D-GS [50]	27.65	0.815	0.220
Bootstrap-GS [15]	28.32	0.823	0.209
3D-GS-UP	27.76	0.818	0.212
Oct-3DGS-UP	27.81	0.818	0.217
3D-GS-B	28.51	0.825	0.207
3D-GS-UB	28.59	0.826	0.207
Oct-3DGS-UB	28.71	0.828	0.211

to enhance performance matrices and alleviate persistent artifacts.

#### 8.2. Results on Different Framework

Since both Octree-UP and Octree-UB are based on the MLP-based Scaffold-GS framework, we additionally trained another version using the traditional Spherical Harmonics-based original 3D-GS to further validate the effectiveness of our methods on the Mip-NeRF360 dataset. The results are provided in Table 4. With the refined bootstrapping pipeline, the improvements in metrics are more pronounced than those achieved by the original Bootstrap-GS pipeline as demonstrated in **3D-GS-B**, where we use the same training configuration of Bootstrap-GS but different bootstrapping implementation. Consistent with our conclusion in Section 4.4, training on Spherical Harmonics better leverages the potential of bootstrapping and upscaling, as each Gaussian primitive can be modified independently without the influence of the color prediction MLP. Note that the results of Bootstrap-GS are trained on our finetuned diffusion models with its original configuration.

#### 8.3. Ablation Study

Our ablation study primarily focuses on the configurations and implementations of the diffusion models within our framework. We present the results of these ablation studies on bootstrapping diffusion models in Table 5.

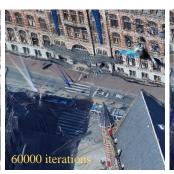
#### 8.3.1. Bootstrapping Ablation Study

In this context, **Octree-UB-raw** refers to the raw implementation of **SDXL-Turbo** without fine-tuning on each specific dataset. While this configuration shows slight overall improvements, the increments are inconsistent across different scenes. For instance, the results for the **Drjohnson** scene exhibit slight degradation, whereas the **Playroom** scene shows significant enhancement.

We also experimented with different textual prompts for conditional generation using the unfinetuned SDXL-Turbo, since after fine-tuning, we can discard the prompt guidance







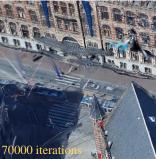


Figure 7. Artifacts of Octree-GS in zoomed-in views compared with different training iterations.

Table 5. Ablation studies on DeepBlending [18] dataset. In the #GS(k)/Mem matrix, entries marked with a "-" indicate that the variations compared to the corresponding original versions are negligible, and #GS(k)/Mem is also not the primary focus of these studies, these metrics do not provide meaningful comparisons.

Dataset		Deep	Blending	
Method Metrics	PSNR (↑)	SSIM (†)	LPIPS (↓)	#GS(k)/Mem
Octree-GS [50]	30.49	0.912	0.241	112/71.7M
Octree-UB	31.23	0.919	0.237	117/75.6M
Octree-UB-raw	30.62	0.913	0.240	116/74.2M
Octree-UB-prompt-1	30.61	0.913	0.240	-
Octree-UB-prompt-2	30.52	0.911	0.241	-
Octree-UB-infer-1	31.23	0.919	0.237	-
Octree-UB-infer-n	31.29	0.920	0.237	-
Octree-UB-var-1	31.16	0.918	0.238	115/73.7M
Octree-UB-var-3	31.17	0.919	0.238	118/75.5M
Octree-UB-uncropped	31.11	0.916	0.239	120/77.1M
Octree-UB-start-15k	31.38	0.921	0.236	122/79.4M
Octree-UB-start-10k	31.49	0.923	0.235	126/80.7M
Octree-UB-loss+	31.31	0.920	0.237	122/79.4M
Octree-UB-de-MLP	31.32	0.919	0.237	-

altogether. In Octree-UB-prompt-1, we used the prompt "professional graph with fine-grained details", similar to the one employed in Bootstrap-GS [15]. This resulted in minimal changes in performance. However, in Octree-UB-prompt-2, where we added each scene's name in the first word of our original prompt, the results exhibited degradation. This underscores the necessity of fine-tuning the bootstrapping diffusion model, as the pretrained posterior of SDXL-Turbo may not align well with our datasets.

Further, we evaluated the impact of different inference time steps. **Octree-UB-infer-1** corresponds to our default Octree-UB configuration, performing inference in a single step. In contrast, **Octree-UB-infer-n** sets the number of inference steps to a fixed value of 100, thereby conducting multiple steps during inference. Although this approach yields a modest performance increase, it requires significantly more computational time. Therefore, a single inference step is sufficient for basic training.

We also tested the effect of varying the number of variants used in bootstrapping. **Octree-UB-var-1** utilizes one variant, while **Octree-UB-var-3** employs three variants. The results indicate that using only one variant may be insufficient to achieve the multi-view alignment required in Bootstrap-GS. Conversely, using three variants introduces excessive disturbance due to the random sampling of camera variants, which adversely affects the performance of the color MLP. Additionally, the uncropped bootstrapping training configuration **Octree-UB-uncropped** proved to be significantly more unstable than our cropped version.

Moreover, initiating bootstrapping at earlier stages of training can enhance performance in normal views, as shown in **Octree-UB-start-15k** and **Octree-UB-start-10k**, where bootstrapping begins at 15,000 and 10,000 iterations, respectively. However, this strategy is not universally beneficial for all scenes in zoomed-in views, particularly for certain scenes in the Mip-NeRF 360 dataset, as illustrated

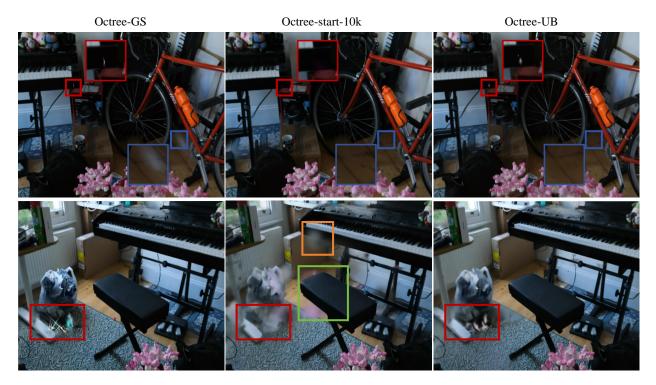


Figure 8. Artifacts of Mip-NeRF360 Bonsai in zoomed-in views compared with different frameworks.

in Figure 8. In Figure 8(a), where the scenes are generally homogeneous, increased bootstrapping yields significantly better results in zoomed-in views compared to the original Octree-GS. However, in Figure 8(b), where the scenes are largely collapsed in the original models, our techniques tend to produce overly smoothed results. In these cases, the diffusion models may lack sufficient information to accurately complete the images, since their finetuning is based on normal views where scenes are better reconstructed, as described in Bootstrap-GS [15].

As a result, due to the averaging effect in the bootstrapping and upscaling loss functions, these scenes appear blurred and over-smoothed. While over-smoothing can mitigate sharp and chaotic noise—which is preferable to having prominent artifacts—the color MLP can exacerbate over-smoothing in regions that originally lacked significant artifacts. Specifically, in Figure 8(b), Octree-UBstart-10k displays over-smoothed artifacts even in normal areas. This phenomenon aligns with our discussion in Section 8.1, where Gaussian primitives modified at early stages become invisible in later stages, and the alterations introduced by the color MLP during bootstrapping lead to oversmoothed artifacts. In addition, we adjusted the loss weight  $\lambda_b = [0.25, 0.2]$  in Octree-UB-loss+ based on Octree-UB, and this adjustment resulted in similar over-smoothed artifacts as those encountered in Octree-UB-start-10k.

Finally, in the configuration denoted as Octree-UB-de-

MLP, we experimented with deactivating the MLP updating during bootstrapping iterations, building upon Octree-UB-start-10k. In this setup, only the features of Gaussian primitives [36] are modified during bootstrapping, effectively eliminating the overly smoothed artifacts in normal regions caused by the color MLP. Although the training of the color MLP is affected and the normal-view matrices are slightly worse than those of Octree-UB-start-15k, we consider this a worthwhile trade-off to achieve more integrated zoomed-in views. However, in our main paper, we did not adopt this strategy because it increases the training time and we aimed to maintain fair comparisons with other frameworks without altering their training strategies.

#### 8.3.2. Upsaling Ablation Study

In this part, we have tried extensive configurations of upscale diffusion models and the loss values, especially on the multi-scale dataset BungeeNeRF. However, from the perspective of matrices, the differences are negligible, as shown in Table 6.

In the configurations Octree-UPx2-l+ and Octree-UPx4-l+, we decoupled the upscaling loss from the scaling factor  $\lambda_u$  and included it as an independent component in the loss function. We observed that the performance in normal views and the properties of Gaussian primitives remained largely unchanged, indicating that zoomed-in views exert minimal influence on normal-scale renderings. However, regarding Gaussian expansion, we found that the up-

Table 6. Ablation studies on the BungeeNeRF [69] scenes.

Dataset		Bung	eeNeRF	
Method Metrics	PSNR (↑)	SSIM $(\uparrow)$	$\text{LPIPS}(\downarrow)$	#GS(k)/Mem
Octree-GS [50]	28.39	0.923	0.088	1474/296.7M
Octree-UPx2	28.43	0.923	0.088	1521/306.2M
Octree-UPx4	28.48	0.924	0.087	1568/315.6M
Octree-UPx2-l+	28.45	0.923	0.088	1523/306.6M
Octree-UPx4-l+	28.43	0.923	0.088	1569/315.8M
Octree-UPx2-var-1	28.41	0.923	0.087	1513/304.5M
Octree-UPx4-var-1	28.46	0.924	0.088	1544/310.8M
Octree-UPx2-var-3	28.49	0.925	0.087	1529/307.9M
Octree-UPx4-var-3	28.52	0.925	0.086	1591/320.4M
Octree-UPx2-uncropped	28.33	0.921	0.090	1545/311.1M
Octree-UPx4-uncropped	28.27	0.921	0.091	1597/321.4M

scaling loss was excessively large even when  $\lambda_u$  was set to a very small value. Consequently, the duplication of Gaussian primitives had a magnitude similar to that of the scaled upscaling loss. As a result, the increase in the number of Gaussian primitives compared to **Octree-UPx2** and **Octree-UPx4** was minimal.

Nonetheless, augmenting the loss in Octree-Scaffold-GS led to more pronounced over-smoothed rendering results than those depicted in Figure 8. We previously discussed the reasons for this in both Section 8.1 and the preceding paragraph. Even with numerous expanded views, most of the zoomed-in regions are not adequately modified during training. An excessively large upscaling loss disrupts the original distribution of the color MLP, thereby affecting the imperceptible Gaussian primitives that were initially not significantly divergent from the overall scene. Conversely, in the absence of a color MLP, we can directly incorporate the upscaling loss without applying a scaling factor.

After adjusting the number of variants used to generate random upscaling cameras, as described in Section 7.6, we observed that increasing the number of constructed variants led to improved results. Since many Gaussian primitives in zoomed-in regions are only trained during the early stages (illustrated in Section 8.1), introducing additional zoomed-in cameras helps to better integrate the entire scene from specific perspectives. Although the variations in evaluation metrics are not significant, the zoomed-in renderings can be substantially improved. The observed variations in the corresponding Gaussian primitives further validate our conclusion. Notice that, here,  $\lambda_u$  is very small and would not affect the overall performance of color MLP.

For uncropped upscaling configurations, namely **Octree-UPx2-uncropped** and **Octree-UPx4-uncropped**, the camera horizons are significantly expanded, resulting in a substantial increase in the expansion of Gaussian primitives.

However, we do not recommend this approach. We found that when upscaling larger images, the diffusion models tend to generate more flexible details that are not permissible under our upscaling constraints. As a result, we observed noticeable performance degradation. Additionally, before performing large-scale upscaling, it is important to ensure that the code does not automatically embed watermarks in the generated images defaulted in **Stable Diffusion Configuration**, as this can also adversely affect the results.

#### **8.4. Per-scene Results**

In Tables 7-14, we list the metrics used in our evaluation in Sec. 5 across all considered methods and scenes.

Table 7. Quantitative results for all scenes in the Tanks&Temples [26] dataset.

Dataset		Truck			Train	
Method Metrics	PSNR(↑)	SSIM(↑)	$LPIPS(\downarrow)$	PSNR(↑)	SSIM(↑)	$LPIPS(\downarrow)$
3D-GS [25]	25.52	0.884	0.142	22.30	0.819	0.201
Mip-Splatting [80]	25.74	0.888	0.142	22.17	0.824	0.199
Scaffold-GS [36]	26.04	0.889	0.131	22.91	0.838	0.181
Boot-GS [15]	26.23	0.891	0.139	23.47	0.835	0.187
Octree-GS [50]	26.24	0.894	0.122	23.11	0.838	0.184
Octree-UP	26.21	0.894	0.122	23.34	0.841	0.182
Octree-UB	26.47	0.896	0.120	23.83	0.846	0.179

Table 8. Quantitative results for all scenes in the DeepBlending [18] dataset.

Dataset		Dr Johnso	n		Playroom	1
Method Metrics	PSNR(↑)	$\text{SSIM}(\uparrow)$	$LPIPS(\downarrow)$	PSNR(↑)	$\text{SSIM}(\uparrow)$	$LPIPS(\downarrow)$
3D-GS [25]	29.09	0.900	0.242	29.83	0.905	0.241
Mip-Splatting [80]	29.08	0.900	0.241	30.03	0.902	0.245
Scaffold-GS [36]	29.73	0.910	0.235	30.83	0.907	0.242
Boot-GS [15]	30.92	0.911	0.231	31.95	0.918	0.232
Octree-GS [50]	29.83	0.909	0.237	31.15	0.914	0.245
Octree-UP	29.94	0.910	0.236	31.37	0.917	0.242
Octree-UB	30.64	0.915	0.234	31.82	0.921	0.240

Table 9. PSNR ( $\uparrow$ ) for all scenes in the Mip-NeRF360 [3] dataset.

Method Scenes	Bicycle	Bonsai	Counter	Flowers	Garden	Kitchen	Room	Stump	Treehill
3D-GS [25]	25.10	32.19	29.22	21.57	27.45	31.62	31.53	26.70	22.46
Mip-Splatting [80]	25.13	32.56	29.30	21.64	27.43	31.48	31.73	26.65	22.60
Scaffold-GS [36]	25.19	33.22	29.99	21.40	27.48	31.77	32.30	26.67	23.08
Boot-GS [15]	26.02	33.31	29.98	21.93	28.34	32.11	32.28	27.68	23.31
Octree-GS [50]	25.24	33.76	30.19	21.46	27.67	31.84	32.51	26.63	23.13
Octree-UP	25.22	33.85	30.27	21.65	27.62	31.96	32.55	26.83	23.31
Octree-UB	25.61	33.98	30.63	22.14	27.93	32.19	32.85	27.72	23.77

Table 10. SSIM (†) for all scenes in the Mip-NeRF360 [3] dataset.

Method Scenes	Bicycle	Bonsai	Counter	Flowers	Garden	Kitchen	Room	Stump	Treehill
3D-GS [25]	0.747	0.947	0.917	0.600	0.861	0.932	0.926	0.773	0.636
Mip-Splatting [80]	0.747	0.948	0.917	0.601	0.861	0.933	0.928	0.772	0.639
Scaffold-GS [36]	0.751	0.952	0.922	0.587	0.853	0.931	0.932	0.767	0.644
Boot-GS [15]	0.763	0.951	0.922	0.608	0.868	0.937	0.935	0.784	0.645
Octree-GS [50]	0.755	0.955	0.925	0.595	0.861	0.933	0.936	0.766	0.641
Octree-UP	0.755	0.956	0.924	0.597	0.861	0.932	0.935	0.769	0.645
Octree-UB	0.759	0.957	0.930	0.604	0.863	0.938	0.938	0.784	0.650

Table 11. LPIPS  $(\downarrow)$  for all scenes in the Mip-NeRF360 [3] dataset.

Method Scenes	Bicycle	Bonsai	Counter	Flowers	Garden	Kitchen	Room	Stump	Treehill
3D-GS [25]	0.243	0.178	0.179	0.345	0.114	0.117	0.196	0.231	0.335
Mip-Splatting [80]	0.245	0.178	0.179	0.347	0.115	0.115	0.192	0.232	0.334
Scaffold-GS [36]	0.247	0.173	0.177	0.359	0.130	0.118	0.183	0.252	0.338
Boot-GS [15]	0.236	0.169	0.174	0.341	0.110	0.114	0.191	0.225	0.327
Octree-GS	0.235	0.164	0.169	0.347	0.116	0.115	0.172	0.250	0.360
Octree-UP	0.236	0.164	0.168	0.345	0.116	0.117	0.173	0.247	0.356
Octree-UB	0.235	0.161	0.165	0.341	0.114	0.113	0.171	0.242	0.354

Table 12. PSNR (↑) for all scenes in the BungeeNeRF [69] dataset.

Method Scenes	Amsterdam	Barcelona	Bilbao	Chicago	Hollywood	l Pompidou	Quebec	Rome
3D-GS [25]	27.75	27.55	28.91	28.27	26.25	27.16	28.86	27.56
Mip-Splatting [28]	28.16	27.72	29.13	28.28	26.59	27.71	29.23	28.33
Scaffold-GS [36]	27.82	28.09	29.20	28.55	26.36	27.72	29.29	28.24
Octree-GS [50]	28.16	28.40	29.39	28.86	26.76	27.46	29.46	28.59
Octree-UPx2	28.14	28.32	29.51	28.94	26.71	27.64	29.51	28.55
Octree-UPx4	28.18	28.43	29.47	29.11	26.71	27.49	29.62	28.68

Table 13. SSIM (†) for all scenes in the BungeeNeRF [69] dataset.

Method Scenes	Amsterdam	Barcelona	Bilbao	Chicago	Hollywood	Pompidou	Quebec	Rome
3D-GS [25]	0.918	0.919	0.918	0.932	0.873	0.919	0.937	0.918
Mip-Splatting [80]	0.918	0.919	0.918	0.930	0.876	0.923	0.938	0.922
Scaffold-GS [36]	0.914	0.923	0.918	0.929	0.866	0.926	0.939	0.924
Octree-GS [50]	0.922	0.928	0.923	0.935	0.884	0.925	0.942	0.930
Octree-UPx2	0.922	0.927	0.924	0.934	0.884	0.927	0.942	0.930
Octree-UPx4	0.922	0.929	0.923	0.936	0.884	0.925	0.944	0.931

Table 14. LPIPS  $(\downarrow)$  for all scenes in the BungeeNeRF [69] dataset.

Method Scenes	Amsterdam	Barcelona	a Bilbao	Chicago	Hollywood	l Pompidou	Quebec	Rome
3D-GS [25]	0.092	0.082	0.092	0.080	0.128	0.090	0.087	0.096
Mip-Splatting [80]	0.094	0.082	0.095	0.081	0.130	0.087	0.087	0.093
Scaffold-GS [36]	0.102	0.078	0.090	0.08	0.157	0.082	0.080	0.087
Our-Scaffold-GS	0.090	0.071	0.091	0.077	0.128	0.089	0.081	0.080
Octree-UPx2	0.091	0.072	0.090	0.077	0.128	0.088	0.081	0.080
Octree-UPx4	0.091	0.070	0.091	0.075	0.128	0.089	0.080	0.079