GaussianObject: High-Quality 3D Object Reconstruction from Four Views with Gaussian Splatting

CHEN YANG*, MoE Key Lab of Artificial Intelligence, AI Institute, SJTU, China
SIKUANG LI*, MoE Key Lab of Artificial Intelligence, AI Institute, SJTU, China
JIEMIN FANG†, Huawei Inc., China
RUOFAN LIANG, University of Toronto, Canada
LINGXI XIE, Huawei Inc., China
XIAOPENG ZHANG, Huawei Inc., China
WEI SHEN‡, MoE Key Lab of Artificial Intelligence, AI Institute, SJTU, China
QI TIAN, Huawei Inc., China

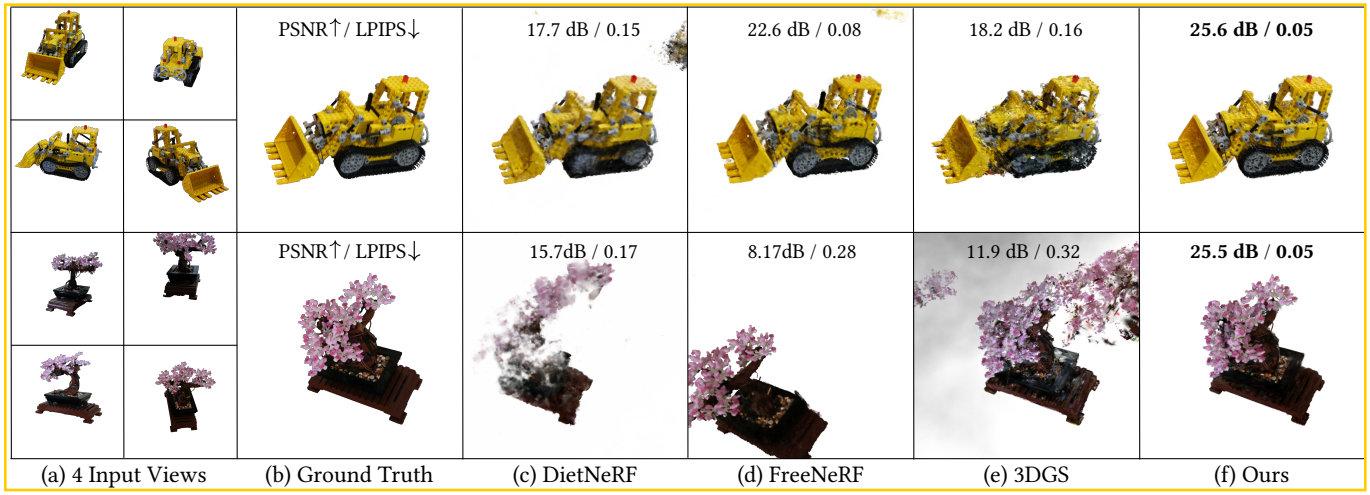

Fig. 1. We introduce GaussianObject, a framework capable of reconstructing high-quality 3D objects from only 4 images with Gaussian splatting. GaussianObject demonstrates superior performance over previous state-of-the-art (SOTA) methods on challenging objects.

Reconstructing and rendering 3D objects from highly sparse views is of critical importance for promoting applications of 3D vision techniques and improving user experience. However, images from sparse views only contain very limited 3D information, leading to two significant challenges: 1) Difficulty in building multi-view consistency as images for matching are too few; 2) Partially omitted or highly compressed object information as view coverage is insufficient. To tackle these challenges, we propose GaussianObject, a framework to represent and render the 3D object with Gaussian splatting that achieves high rendering quality with only 4 input images. We first introduce techniques of visual hull and floater elimination, which explicitly inject structure priors into the initial optimization process to help build multi-view consistency, yielding a coarse 3D Gaussian representation. Then we construct a Gaussian repair model based on diffusion models to supplement the omitted object information, where Gaussians are further refined. We design a self-generating strategy to obtain image pairs for training the repair model. We further design a COLMAP-free variant, where pre-given accurate camera poses are not required, which achieves competitive quality and facilitates wider applications. GaussianObject is evaluated on several challenging datasets, including MipNeRF360, OmniObject3D, OpenIllumination, and our-collected unposed images, achieving superior performance from only four views and significantly outperforming previous SOTA methods. Our demo is available at *https://gaussianobject.github.io/*, and the code has been released at *https://github.com/GaussianObject/GaussianObject*.

CCS Concepts: • **Computing methodologies** → **Reconstruction**; **Rendering**; **Point-based models**.

Additional Key Words and Phrases: Sparse view reconstruction, 3D Gaussian Splatting, ControlNet, Visual hull, Novel view synthesis

ACM Reference Format:
Chen Yang, Sikuang Li, Jiemin Fang, Ruofan Liang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. 2024. GaussianObject: High-Quality 3D Object

*Equal contributions.
†Project lead.
‡Corresponding author.

Authors' addresses: Chen Yang, MoE Key Lab of Artificial Intelligence, AI Institute, SJTU, Shanghai, China, ycyangchen@sjtu.edu.cn; Sikuang Li, MoE Key Lab of Artificial Intelligence, AI Institute, SJTU, Shanghai, China, uranusits@sjtu.edu.cn; Jiemin Fang, Huawei Inc., Wuhan, China, jaminfong@gmail.com; Ruofan Liang, University of Toronto, Toronto, Canada, ruofan@cs.toronto.edu; Lingxi Xie, Huawei Inc., Beijing, China, 198808xc@gmail.com; Xiaopeng Zhang, Huawei Inc., Shanghai, China, zxphistory@gmail.com; Wei Shen, MoE Key Lab of Artificial Intelligence, AI Institute, SJTU, Shanghai, China, wei.shen@sjtu.edu.cn; Qi Tian, Huawei Inc., Shenzhen, China, tian.qi1@huawei.com.

2024. ACM 0730-0301/2024/12-ART
https://doi.org/10.1145/3687759

Reconstruction from Four Views with Gaussian Splatting. *ACM Trans. Graph.* 43, 6 (December 2024), 28 pages. https://doi.org/10.1145/3687759

1 INTRODUCTION

Reconstructing and rendering 3D objects from 2D images has been a long-standing and important topic, which plays critical roles in a vast range of real-life applications. One key factor that impedes users, especially ones without expert knowledge, from widely using these techniques is that usually dozens of multi-view images need to be captured, which is cumbersome and sometimes impractical. Efficiently reconstructing high-quality 3D objects from highly sparse captured images is of great value for expediting downstream applications such as 3D asset creation for game/movie production and AR/VR products.

In recent years, a series of methods [Guangcong et al. 2023; Jain et al. 2021; Niemeyer et al. 2022; Shi et al. 2024b; Song et al. 2023b; Yang et al. 2023; Zhou and Tulsiani 2023; Zhu et al. 2024] have been proposed to reduce reliance on dense captures. However, it is still challenging to produce high-quality 3D objects when the views become **extremely sparse**, *e.g.* only 4 images in a 360° range, as shown in Fig. 1. We delve into the task of sparse-view reconstruction and discover two main challenges behind it. The first one lies in the difficulty of building multi-view consistency from highly sparse input. The 3D representation is easy to overfit the input images and degrades into fragmented pixel patches of training views without reasonable structures. The other challenge is that with sparse captures in a 360° range, some content of the object can be inevitably omitted or severely compressed when observed from extreme views[1]. The omitted or compressed information is impossible or hard to be reconstructed in 3D only from the input images.

To tackle the aforementioned challenges, we introduce GaussianObject, a novel framework designed to reconstruct high-quality 3D objects from as few as 4 input images. We choose 3D Gaussian splatting (3DGS) [Kerbl et al. 2023] as the basic representation as it is fast and, more importantly, explicit enough. Benefiting from its point-like structure, we design several techniques for introducing object structure priors, *e.g.* the basic/rough geometry of the object, to help build multi-view consistency, including visual hull [Laurentini 1994] to locate Gaussians within the object outline and floater elimination to remove outliers. To erase artifacts caused by omitted or highly compressed object information, we propose a Gaussian repair model driven by 2D large diffusion models [Rombach et al. 2022], translating corrupted rendered images into high-fidelity ones. As normal diffusion models lack the ability to repair corrupted images, we design self-generating strategies to construct image pairs to tune the diffusion models, including rendering images from leave-one-out training models and adding 3D noises to Gaussian attributes. Images generated from the repair model can be used to refine the 3D Gaussians optimized with structure priors, where the rendering quality can be further improved. To further extend GaussianObject to practical applications, we introduce a COLMAP-free variant of GaussianObject (CF-GaussianObject), which achieves competitive reconstruction performance on challenging datasets with only four input images without inputting accurate camera parameters.

Our contributions are summarized as follows:

- We optimize 3D Gaussians from highly sparse views using explicit structure priors, introducing techniques of visual hull for initialization and floater elimination for training.
- We propose a Gaussian repair model based on diffusion models to remove artifacts caused by omitted or highly compressed information, where the rendering quality can be further improved.
- The overall framework GaussianObject consistently outperforms current SOTA methods on several challenging real-world datasets, both qualitatively and quantitatively. A COLMAP-free variant is further presented for wider applications, weakening the requirement of accurate camera poses.

2 RELATED WORK

Vanilla NeRF struggles in sparse settings. Techniques like Deng et al. [2022]; Roessle et al. [2022]; Somraj et al. [2024, 2023]; Somraj and Soundararajan [2023] use Structure from Motion (SfM) [Schönberger and Frahm 2016] derived visibility or depth and mainly focus on closely aligned views. Xu et al. [2022] uses ground truth depth maps, which are costly to obtain in real-world images. Some methods [Guangcong et al. 2023; Song et al. 2023b] estimate depths with monocular depth estimation models [Ranftl et al. 2021, 2022] or sensors, but these are often too coarse. Jain et al. [2021] uses a vision-language model [Radford et al. 2021] for unseen view rendering, but the semantic consistency is too high-level to guide low-level reconstruction. Shi et al. [2024b] combines a deep image prior with factorized NeRF, effectively capturing overall appearance but missing fine details in input views. Priors based on information theory [Kim et al. 2022], continuity [Niemeyer et al. 2022], symmetry [Seo et al. 2023], and frequency regularization [Song et al. 2023a; Yang et al. 2023] are only effective for specific scenarios, limiting their further applications. Besides, there are some methods [Jang and Agapito 2024; Jiang et al. 2024; Xu et al. 2024c; Zou et al. 2024] that employ Vision Transformer (ViT) [Dosovitskiy et al. 2021] to reduce the requirements for constructing NeRFs and Gaussians.

The recent progress in diffusion models has spurred notable advancements in 3D applications. Dreamfusion [Poole et al. 2023] proposes Score Distillation Sampling (SDS) for distilling NeRFs with 2D priors from a pre-trained diffusion model for 3D object generation from text prompts. It has been further refined for text-to-3D [Chen et al. 2023; Lin et al. 2023; Metzer et al. 2023; Shi et al. 2024a; Tang et al. 2024b; Wang et al. 2023a,b; Yi et al. 2024] and 3D/4D editing [Haque et al. 2023; Shao et al. 2024] by various studies, demonstrating the versatility of 2D diffusion models in 3D contexts. Burgess et al. [2024]; Chan et al. [2023]; Liu et al. [2023c]; Müller et al. [2024]; Pan et al. [2024]; Zhu and Zhuang [2024] have adapted these methods for 3D generation and view synthesis from a single image, while they often have strict input requirements and can produce overly saturated images. In sparse reconstruction, approaches like DiffusioNeRF [Wynn and Turmukhambetov 2023], SparseFusion [Zhou and Tulsiani 2023], Deceptive-NeRF [Liu et al. 2023b], ReconFusion [Wu et al. 2024] and CAT3D [Gao et al. 2024] integrate diffusion models with NeRFs. Recently, Large reconstruction

[1] When the view is orthogonal to the surface of the object, the observed information attached to the surface can be largely preserved; On the contrary, the information will be severely compressed.

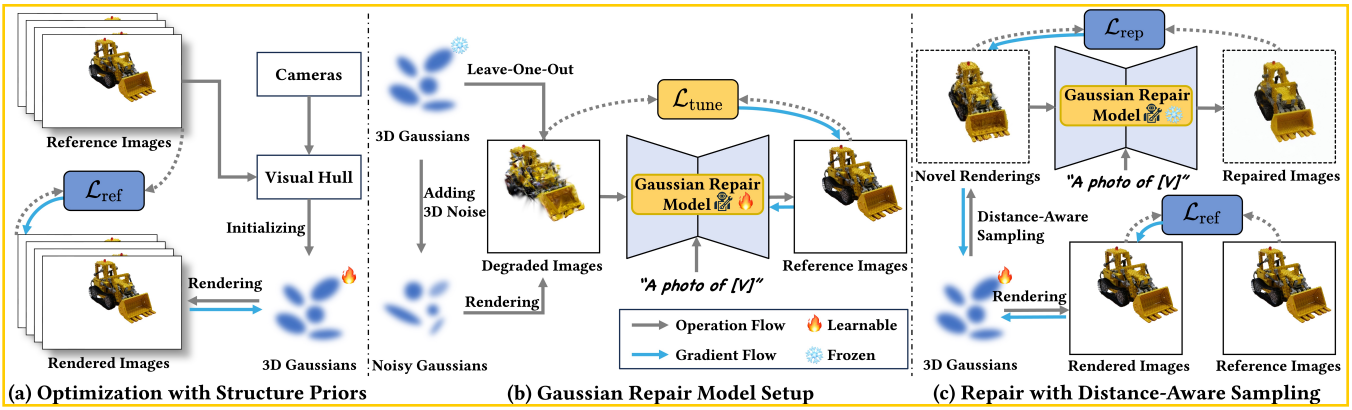

Fig. 2. Overview of GaussianObject. (a) We initialize 3D Gaussians by constructing a visual hull with camera parameters and masked images, which are optimized with $\mathcal{L}_{\text{ref}}$ and refined through floater elimination. (b) We use a novel 'leave-one-out' strategy and add 3D noise to Gaussians to generate corrupted Gaussian renderings. These renderings, paired with their corresponding reference images, facilitate the training of the Gaussian repair model employing $\mathcal{L}_{\text{tune}}$. For details please refer to Fig. 3. (c) Once trained, the Gaussian repair model is frozen and used to correct views that need to be rectified. These views are identified through distance-aware sampling. The repaired images and reference images are used to further optimize 3D Gaussians with $\mathcal{L}_{\text{rep}}$ and $\mathcal{L}_{\text{ref}}$.

models (LRMs) [Hong et al. 2024; Li et al. 2024; Tang et al. 2024a; Wang et al. 2024b; Wei et al. 2024; Weng et al. 2023; Xu et al. 2024a,b; Zhang et al. 2024] also achieve 3D reconstruction from highly sparse views. Though effective in generating images fast, these methods encounter issues with large pretraining, strict requirements on view distribution and object location, and difficulty in handling real-world captures.

While 3DGS shows strong power in novel view synthesis, it struggles with sparse 360° views similar to NeRF. Inspired by few-shot NeRFs, methods [Charatan et al. 2024; Chung et al. 2023; Paliwal et al. 2024; Xiong et al. 2023; Zhu et al. 2024] have been developed for sparse 360° reconstruction. However, they still severely rely on the SfM points. Our GaussianObject proposes structure-prior-aided Gaussian initialization to tackle this issue, drastically reducing the required input views to only 4, a significant improvement compared with over 20 views required by FSGS [Zhu et al. 2024].

3 METHOD

The subsequent sections detail the methodology: Sec. 3.1 reviews foundational techniques; Sec. 3.2 introduces our overall framework; Sec. 3.3 describes how we apply the structure priors for initial optimization; Sec. 3.4 details the setup of our Gaussian repair model; Sec. 3.5 illustrates the repair of 3D Gaussians using this model and Sec. 3.6 elucidates the COLMAP-free version of GaussianObject. To facilitate a better understanding, all key mathematical symbols and their corresponding meanings are listed in Table 1.

Table 1. List of Key Mathematical Symbols

Symbol	Meaning
$X^{\text{ref}} = \{x_i\}_{i=1}^{N}$	Reference images
$K^{\text{ref}} = \{k_i\}_{i=1}^{N}$	Intrinsics of X^{ref}
$\hat{K}^{\text{ref}}$	Estimated intrinsics of X^{ref}
$\hat{K}$	Estimated shared intrinsics of X^{ref}
$\Pi^{\text{ref}} = \{\pi_i\}_{i=1}^{N}$	Extrinsics of X^{ref}
Π^{nov}	Extrinsics of viewpoints in repair path
$\hat{\Pi}^{\text{ref}}$	Estimated extrinsics of X^{ref}
$M^{\text{ref}} = \{m_i\}_{i=1}^{N}$	Masks of X^{ref}
μ	Center location of Gaussian
q	Rotation quaternion of Gaussian
s	Scale vector of Gaussian
σ	Opacity of Gaussian
sh	Spherical harmonic coefficients of Gaussian
$\mathcal{G}_c$	Coarse 3D Gaussians
$\mathcal{R}$	Diffusion based Gaussian repair model
$\mathcal{E}$	Latent diffusion encoder of $\mathcal{R}$
$\mathcal{D}$	Latent diffusion decoder of $\mathcal{R}$
x'	Degraded rendering
$\hat{x}$	Image repaired by $\mathcal{R}$
ϵ_s	3D Noise added to attributes of $\mathcal{G}_c$
ϵ	2D Gaussian noise for fine-tuning
ϵ_θ	2D Noise predicted by $\mathcal{R}$
c^{tex}	Object-specific language prompt
$\mathcal{P}$	Coarse point cloud predicted by DUSt3R

3.1 Preliminary

3D Gaussian Splatting. 3D Gaussian Splatting [Kerbl et al. 2023] represents 3D scenes with 3D Gaussians. Each 3D Gaussian is composed of the center location μ, rotation quaternion q, scaling vector s, opacity σ, and spherical harmonic (SH) coefficients sh. Thus, a scene is parameterized as a set of Gaussians $\mathcal{G} = \{G_i : \mu_i, q_i, s_i, \sigma_i, sh_i\}_{i=1}^{P}$.

ControlNet. Diffusion models are generative models that sample from a data distribution $q(X_0)$, beginning with Gaussian noise ϵ and using various sampling schedulers. They operate by reversing a discrete-time stochastic noise addition process $\{X_t\}_{t=0}^{T}$ with a diffusion model $p_\theta(X_{t-1}|X_t)$ trained to approximate $q(X_{t-1}|X_t)$, where $t \in [0, T]$ is the noise level and θ is the learnable parameters. Substituting X_0 with its latent code Z_0 from a Variational Autoencoder (VAE) [Kingma and Welling 2014] leads to the development of Latent Diffusion Models (LDM) [Rombach et al. 2022]. ControlNet [Zhang et al. 2023a] further enhances the generative process with additional

image conditioning by integrating a network structure similar to the diffusion model, optimized with the loss function:

$$\mathcal{L}_{Cond} = \mathbb{E}_{Z_0,t,\epsilon}[\|\epsilon_\theta(\sqrt{\bar{\alpha}_t}Z_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t, c^{\text{tex}}, c^{\text{img}}) - \epsilon\|_2^2], \quad (1)$$

where c^{tex} and c^{img} denote the text and image conditioning respectively, and ϵ_θ is the Gaussian noise inferred by the diffusion model with parameter θ, $\bar{\alpha}_{1:T} \in (0,1]^T$ is a decreasing sequence associated with the noise-adding process.

3.2 Overall Framework

Given a sparse collection of N reference images $X^{\text{ref}} = \{x_i\}_{i=1}^N$, captured within a 360° range and encompassing one object, along with the corresponding camera intrinsics[2] $K^{\text{ref}} = \{k_i\}_{i=1}^N$, extrinsics $\Pi^{\text{ref}} = \{\pi_i\}_{i=1}^N$ and masks $M^{\text{ref}} = \{m_i\}_{i=1}^N$ of the object, our target is to obtain a 3D representation $\mathcal{G}$, which can achieve photo-realistic rendering $x = \mathcal{G}(\pi|\{x_i, \pi_i, m_i\}_{i=1}^N)$ from any viewpoint. To achieve this, we employ the 3DGS model for its simplicity for structure priors embedding and fast rendering capabilities. The process begins with initializing 3D Gaussians using a visual hull [Laurentini 1994], followed by optimization with floater elimination, enhancing the structure of Gaussians. Then we design self-generating strategies to supply sufficient image pairs for constructing a Gaussian repair model, which is used to rectify incomplete object information. The overall framework is shown in Fig. 2.

3.3 Initial Optimization with Structure Priors

Sparse views, especially for only 4 images, provide limited 3D information for reconstruction. In this case, SfM points, which are the key for 3DGS initialization, are often absent. Besides, insufficient multi-view consistency leads to ambiguity among shape and appearance, resulting in many floaters during reconstruction. We propose two techniques to initially optimize the 3D Gaussian representation, which take full advantage of structure priors from the limited views and result in a satisfactory outline of the object.

Initialization with Visual Hull. To better leverage object structure information from limited reference images, we utilize the view frustums and object masks to create a visual hull as a geometric scaffold for initializing our 3D Gaussians. Compared with the limited number of SfM points in extremely sparse settings, the visual hull provides more structure priors that help build multiview consistency by excluding unreasonable Gaussian distributions. The cost of the visual hull is just several masks derived from sparse 360° images, which can be easily acquired using current segmentation models such as SAM [Kirillov et al. 2023]. Specifically, points are randomly initialized within the visual hull using rejection sampling: we project uniformly sampled random 3D points onto image planes and retain those within the intersection of all image-space masks. Point colors are averaged from bilinearly interpolated pixel colors across reference image projections. Then we transform these 3D points into 3D Gaussians. For each point, we assign its position as μ and convert its color into sh. The mean distance between adjacent points forms the scale s, while the rotation q is set to a unit quaternion as default. The opacity σ is initialized to a constant value. This initialization strategy relies on the initial masks. Despite potential inaccuracies in these masks or unrepresented concavities by the visual hull, we observed that subsequent optimization processes reliably yield high-quality reconstructions.

Floater Elimination. While the visual hull builds a coarse estimation of the object geometry, it often contains regions that do not belong to the object due to the inadequate coverage of reference images. These regions usually appear to be floaters, damaging the quality of novel view synthesis. These floaters are problematic as the optimization process struggles to adjust them due to insufficient observational data regarding their position and appearance.

To mitigate this issue, we utilize the statistical distribution of distances among the 3D Gaussians to distinguish the primary object and the floaters. This is implemented by the K-Nearest Neighbors (KNN) algorithm, which calculates the average distance to the nearest $\sqrt{P}$ Gaussians for each element in $\mathcal{G}_c$. We then establish a normative range by computing the mean and standard deviation of these distances. Based on statistical analysis, we exclude Gaussians whose mean neighbor distances exceed the adaptive threshold $\tau = \text{mean} + \lambda_e \text{std}$. This thresholding process is repeated periodically throughout optimization, where λ_e is linearly decreased to 0 to refine the scene representation progressively.

Initial Optimization The optimization of $\mathcal{G}_c$ incorporates color, mask, and monocular depth losses. The color loss combines L1 and D-SSIM losses from 3D Gaussian Splatting:

$$\mathcal{L}_1 = \|x - x^{\text{ref}}\|_1, \quad \mathcal{L}_{\text{D-SSIM}} = 1 - \text{SSIM}(x, x^{\text{ref}}), \quad (2)$$

where x is the rendering and x^{ref} is the corresponding reference image. A binary cross entropy (BCE) loss [Jadon 2020] is applied as mask loss:

$$\mathcal{L}_{\text{m}} = -(m^{\text{ref}} \log m + (1 - m^{\text{ref}}) \log(1 - m)), \quad (3)$$

where m denotes the object mask. A shift and scale invariant depth loss is utilized to guide geometry:

$$\mathcal{L}_{\text{d}} = \|D^* - D^*_{\text{pred}}\|_1, \quad (4)$$

where D^* and D^*_{pred} are per-frame rendered depths and monocularly estimated depths [Bhat et al. 2023] respectively. The depth values are computed following a normalization strategy [Ranftl et al. 2020]:

$$D^* = \frac{D - \text{median}(D)}{\frac{1}{M}\sum_{i=1}^{M} |D - \text{median}(D)|}, \quad (5)$$

where M denotes the number of valid pixels. The overall loss combines these components:

$$\mathcal{L}_{\text{ref}} = (1 - \lambda_{\text{SSIM}})\mathcal{L}_1 + \lambda_{\text{SSIM}}\mathcal{L}_{\text{D-SSIM}} + \lambda_{\text{m}}\mathcal{L}_{\text{m}} + \lambda_{\text{d}}\mathcal{L}_{\text{d}}, \quad (6)$$

where λ_{SSIM}, λ_{m}, and λ_{d} control the magnitude of each term. Thanks to the efficient initialization, our training speed is remarkably fast. It only takes 1 minute to train a coarse Gaussian representation $\mathcal{G}_c$ at a resolution of 779 × 520.

3.4 Gaussian Repair Model Setup

Combining visual hull initialization and floater elimination significantly enhances 3DGS performance for NVS in sparse 360° contexts. While the fidelity of our reconstruction is generally passable, $\mathcal{G}_c$

[2] Given that the camera intrinsics are known and fixed, we exclude them from the rendering function for simplicity.

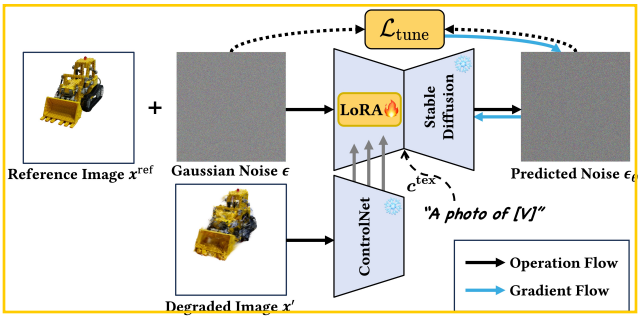

Fig. 3. Illustration of Gaussian repair model setup. First, we add Gaussian noise ϵ to a reference image x^{ref} to form a noisy image. Next, this noisy image along with x^{ref}'s corresponding degraded image x' are passed to a pre-trained fixed ControlNet with learnable LoRA layers to predict a noise distribution ϵ_θ. We use the differences among ϵ and ϵ_θ to fine-tune the parameters in LoRA layers.

still suffers in regions that are poorly observed, regions with occlusion, or even unobserved regions. These challenges loom over the completeness of the reconstruction, like the sword of Damocles.

To mitigate these issues, we introduce a Gaussian repair model $\mathcal{R}$ designed to correct the aberrant distribution of $\mathcal{G}_c$. Our $\mathcal{R}$ takes corrupted rendered images $x'(\mathcal{G}_c, \pi^{\text{nov}})$ as input and outputs photo-realistic and high-fidelity images $\hat{x}$. This image repair capability can be used to refine the 3D Gaussians, leading to learning better structure and appearance details.

Sufficient data pairs are essential for training $\mathcal{R}$ but are rare in existing datasets. To this end, we adopt two main strategies for generating adequate image pairs, *i.e.*, **leave-one-out training** and **adding 3D noises**. For leave-one-out training, we build N subsets from the N input images, each containing $N-1$ reference images and 1 left-out image x^{out}. Then we train N 3DGS models with reference images of these subsets, termed as $\{\mathcal{G}_c^i\}_{i=0}^{N-1}$. After specific iterations, we use the left-out image x^{out} to continue training each Gaussian model $\{\mathcal{G}_c^i\}_{i=0}^{N-1}$ into $\{\hat{\mathcal{G}}_c^i\}_{i=0}^{N-1}$. Throughout this process, the rendered images from the left-out view at different iterations are stored to form the image pairs along with left-out image x^{out} for training the repair model. Note that training these left-out models costs little, with less than N minutes in total. The other strategy is to add 3D noises ϵ_s onto Gaussian attributes. The ϵ_s are derived from the mean μ_Δ and variance σ_Δ of attribute differences between $\{\mathcal{G}_c^i\}_{i=0}^{N-1}$ and $\{\hat{\mathcal{G}}_c^i\}_{i=0}^{N-1}$. This allows us to render more degraded images $x'(\mathcal{G}_c(\epsilon_s), \pi^{\text{ref}})$ at all reference views from the created noisy Gaussians, resulting in extensive image pairs (X', X^{ref}).

We inject LoRA weights and fine-tune a pre-trained ControlNet [Zhang et al. 2023b] using the generated image pairs as our Gaussian repair model. The training procedure is shown in Fig. 3. The loss function, based on Eq. 1, is defined as:

$$\mathcal{L}_{\text{tune}} = \mathbb{E}_{x^{\text{ref}},t,\epsilon,x'}\left[\|(\epsilon_\theta(x_t^{\text{ref}}, t, x', c^{\text{tex}}) - \epsilon)\|_2^2\right], \tag{7}$$

where c^{tex} denotes an object-specific language prompt, defined as "a photo of [V]," as per Dreambooth [Ruiz et al. 2023]. Specifically, we inject LoRA layers into the text encoder, image condition branch, and U-Net for fine-tuning. Please refer to the Appendix for details.

3.5 Gaussian Repair with Distance-Aware Sampling

After training $\mathcal{R}$, we distill its target object priors into $\mathcal{G}_c$ to refine its rendering quality. The object information near the reference views is abundant. This observation motivates designing distance as a criterion in identifying views that need rectification, leading to distance-aware sampling.

Specifically, we establish an elliptical path aligned with the training views and focus on a central point. Arcs near Π^{ref}, where we assume $\mathcal{G}_c$ renders high-quality images, form the reference path. The other arcs, yielding renderings, need to be rectified and define the repair path, as depicted in Fig. 4. In each iteration, novel viewpoints, $\pi_j \in \Pi^{\text{nov}}$, are randomly sampled among the repair path. For each π_j, we render the corresponding image $x_j(\mathcal{G}_c, \pi_j)$, encode it to be $\mathcal{E}(x_j)$ by the latent diffusion encoder $\mathcal{E}$ and pass $\mathcal{E}(x_j)$ to the image conditioning branch of $\mathcal{R}$. Simultaneously, a cloned $\mathcal{E}(x_j)$ is disturbed into a noisy latent z_t:

$$z_t = \sqrt{\bar{\alpha}_t}\mathcal{E}(x_j) + \sqrt{1-\bar{\alpha}_t}\epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, I), t \in [0, T] \tag{8}$$

which is similar to SDEdit [Meng et al. 2022]. We then generate a sample $\hat{x}_j$ from $\mathcal{R}$ by running DDIM sampling [Song et al. 2021] over $k = \lfloor 50 \cdot \frac{t}{T} \rfloor$ steps and forwarding the diffusion decoder $\mathcal{D}$:

$$\hat{x}_j = \mathcal{D}(\text{DDIM}(z_t, \mathcal{E}(x_j))), \tag{9}$$

where $\mathcal{E}$ and $\mathcal{D}$ are from the VAE model used by the diffusion model. The distances from π_j to Π^{ref} is used to weight the reliability of $\hat{x}_j$, guiding the optimization with a loss function:

$$\mathcal{L}_{\text{rep}} = \mathbb{E}_{\pi_j,t}\left[w(t)\lambda(\pi_j)\left(\|x_j - \hat{x}_j\|_1 + \|x_j - \hat{x}_j\|_2 + L_p(x_j, \hat{x}_j)\right)\right],$$
$$\text{where } \lambda(\pi_j) = \frac{2 \cdot \min_{i=1}^{N}(\|\pi_j - \pi_i\|_2)}{d_{\max}}. \tag{10}$$

Here, L_p denotes the perceptual similarity metric LPIPS [Zhang et al. 2018], $w(t)$ is a noise-level modulated weighting function from DreamFusion [Poole et al. 2023], $\lambda(\pi_j)$ denotes a distance-based weighting function, and $d_{\max}$ is the maximal distance among neighboring reference viewpoints. To ensure coherence between 3D Gaussians and reference images, we continue training $\mathcal{G}_c$ with $\mathcal{L}_{\text{ref}}$ during the whole Gaussian repair procedure.

3.6 COLMAP-Free GaussianObject (CF-GaussianObject)

Current SOTA sparse view reconstruction methods rely on precise camera parameters, including intrinsics and poses, obtained through an SfM pipeline with dense input, limiting their usability in daily applications. This process can be cumbersome and unreliable in sparse-view scenarios where matched features are insufficient for accurate reconstruction.

To overcome this limitation, we introduce an advanced sparse matching model, DUSt3R [Wang et al. 2024a], into GaussianObject to enable COLMAP-free sparse 360° reconstruction. Given reference input images X^{ref}, DUSt3R is formulated as:

$$\mathcal{P}, \hat{\Pi}^{\text{ref}}, \hat{K}^{\text{ref}} = \text{DUSt3R}(X^{\text{ref}}), \tag{11}$$

where $\mathcal{P}$ is an estimated coarse point cloud of the scene, and $\hat{\Pi}^{\text{ref}}$, $\hat{K}^{\text{ref}}$ are the predicted camera poses and intrinsics of X^{ref}, respectively. For CF-GaussianObject, we modify the intrinsic recovery module within DUSt3R, allowing $x_i \in X^{\text{ref}}$ to share the same intrinsic $\hat{K}$. This adaption enables the retrieval of $\mathcal{P}$, $\hat{\Pi}^{\text{ref}}$, and $\hat{K}$. Besides,

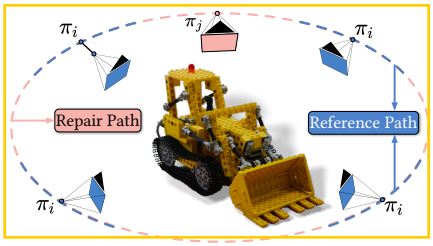

Fig. 4. Illustration of our distance-aware sampling. Blue and red indicate the reference and repair path, respectively.

we apply structural priors with a visual hull to $\mathcal{P}$ to initialize 3D Gaussians. After initialization, we optimize $\hat{\Pi}^{\text{ref}}$ and the initialized 3D Gaussians using X^{ref} and depth maps rendered from $\mathcal{P}$ simultaneously. Besides, we introduce a regularization loss to constrain deviations from $\hat{\Pi}^{\text{ref}}$, enhancing the robustness of the optimization. After optimization, the 3D Gaussians and camera parameters are used for constructing the Gaussian repair model and Gaussian repairing process as described in Sec. 3.4 and Sec. 3.5. Refer to the Appendix for more details.

4 EXPERIMENTS

4.1 Implementation Details

Our framework, illustrated in Fig. 2, is based on 3DGS [Kerbl et al. 2023] and threestudio [Guo et al. 2023]. The 3DGS model is trained for 10k iterations in the initial optimization, with periodic floater elimination every 500 iterations. The monocular depth for $\mathcal{L}_{\text{d}}$ is predicted by ZoeDepth [Bhat et al. 2023]. We use a ControlNet-Tile [Zhang et al. 2023b] model based on stable diffusion v1.5 [Rombach et al. 2022] as our repair model's backbone. LoRA [Hu et al. 2022] weights, injected into the text-encoder and transformer blocks using minLoRA [Chang 2023], are trained for 1800 steps at a LoRA rank of 64 and a learning rate of 10^{-3}. $\mathcal{G}_{\text{c}}$ is trained for another 4k iterations during distance-aware sampling. For the first 2800 iterations, optimization involves both a reference image and a repaired novel view image, with the weight of $\mathcal{L}_{\text{rep}}$ progressively decayed from 1.0 to 0.1. The final 1200-step training only involves reference views. The whole process of GaussianObject takes about 30 minutes on a GeForce RTX 3090 GPU for 4 input images at a 779 × 520 resolution. For more details, please refer to the Appendix.

4.2 Datasets

We evaluate GaussianObject on three datasets suited for sparse-view 360° object reconstruction with varying input views, including Mip-NeRF360 [Barron et al. 2021], OmniObject3D [Wu et al. 2023], and OpenIllumination [Liu et al. 2023a]. Additionally, we use an iPhone 13 to capture four views of some daily-life objects to show the COLMAP-free performance. SAM [Kirillov et al. 2023] is used to obtain masks of the target objects.

4.3 Evaluation

Sparse 360° Reconstruction Performance. We evaluate the performance of GaussianObject against several reconstruction baselines, including the vanilla 3DGS [Kerbl et al. 2023] with random initialization and DVGO [Sun et al. 2022], and various few-view reconstruction models on the three datasets. Compared methods of RegNeRF [Niemeyer et al. 2022], DietNeRF [Jain et al. 2021], SparseNeRF [Guangcong et al. 2023], and ZeroRF [Shi et al. 2024b] utilize a variety of regularization techniques. Besides, FSGS [Zhu et al. 2024] is also built upon Gaussian splatting with SfM-point initialization. Note that we supply extra SfM points to FSGS so that it can work with the highly sparse 360° setting. Since camera pose estimation often suffers from scale and positional errors compared to ground truth, we adopt the evaluation used for COLMAP-free methods under dense view settings [Fu et al. 2024; Wang et al. 2021]. All models are trained using publicly released codes.

Table 2 and 3 present the view-synthesis performance of GaussianObject compared to existing methods on the MipNeRF360, OmniObject3D, and OpenIllumination datasets. Experiments show that GaussianObject consistently achieves SOTA results in all datasets, especially in the perceptual quality – LPIPS. Although GaussianObject is designed to address extremely sparse input views, it still outperforms other methods with more input views, *i.e.* 6 and 9, further proving the effectiveness. Notably, GaussianObject excels with as few as 4 views and significantly improves LPIPS over FSGS from 0.0951 to 0.0498 on MipNeRF360. This improvement is critical, as LPIPS is a key indicator of perceptual quality [Park et al. 2021].

Fig. 5 and Fig. 6 illustrate rendering results of various methods across different datasets with only 4 input views. We observe that GaussianObject achieves significantly better visual quality and fidelity than the competing models. We find that implicit representation based methods and random initialized 3DGS fail in extremely sparse settings, typically reconstructing objects as fragmented pixel patches. This confirms the effectiveness of integrating structure priors with explicit representations. Although ZeroRF exhibits competitive PSNR and SSIM on OpenIllumination, its renderings are blurred and lack details, as shown in Fig. 6. In contrast, GaussianObject demonstrates fine-detailed reconstruction. This superior perceptual quality highlights the effectiveness of the Gaussian repair model. It is highly suggested to refer to comprehensive video comparisons included in supplementary materials.

Comparison with LRMs. We further compare GaussianObject to recently popular LRM-like feed-forward reconstruction methods, *i.e.* LGM [Tang et al. 2024a] and TriplaneGaussian (TGS) [Zou et al. 2024] which are publicly available. The comparisons are shown in Table 4 on the challenging MipNeRF360 dataset. Given that TriplaneGaussian accommodates only a single image input, we feed it with frontal views of objects. LGM requires placing the target object at the world coordinate origin with cameras oriented towards it at an elevation of 0° and azimuths of 0°, 90°, 180°, and 270°. Therefore, we report two versions of LGM – LGM-4 which uses four sparse captures as input views directly, and LGM-1 which uses MVDream [Shi et al. 2024a] to generate images that comply with LGM's setup requirements following its original manner. Results show that the strict requirements among input views significantly hinder the sparse reconstruction performance of LRM-like models with in-the-wild captures. In contrast, GaussianObject does not require extensive pre-training, has no restrictions on input views, and can reconstruct any complex object in daily life.

Fig. 5. Qualitative examples on the MipNeRF360 and OmniObject3D dataset with 4 input views. Many methods fail to reach a coherent 3D representation, resulting in floaters and disjoint pixel patches. A pure white image indicates a total miss of the object by the corresponding method, usually caused by overfitting the input images.

Table 2. Comparisons with varying input views. LPIPS* = LPIPS × 10^2 throughout this paper. Best results are highlighted as 1st, 2nd and 3rd.

	Method	4-view			6-view			9-view		
		LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑
MipNeRF360	DVGO [Sun et al. 2022]	24.43	14.39	0.7912	26.67	14.30	0.7676	25.66	14.74	0.7842
	3DGS [Kerbl et al. 2023]	10.80	20.31	0.8991	8.38	22.12	0.9134	6.42	24.29	0.9331
	DietNeRF [Jain et al. 2021]	11.17	18.90	0.8971	6.96	22.03	0.9286	5.85	23.55	0.9424
	RegNeRF [Niemeyer et al. 2022]	20.44	13.59	0.8476	20.72	13.41	0.8418	19.70	13.68	0.8517
	FreeNeRF [Yang et al. 2023]	16.83	13.71	0.8534	6.84	22.26	0.9332	5.51	27.66	0.9485
	SparseNeRF [Guangcong et al. 2023]	17.76	12.83	0.8454	19.74	13.42	0.8316	21.56	14.36	0.8235
	ZeroRF [Shi et al. 2024b]	19.88	14.17	0.8188	8.31	24.14	0.9211	5.34	27.78	0.9460
	FSGS [Zhu et al. 2024]	9.51	21.07	0.9097	7.69	22.68	0.9264	6.06	25.31	0.9397
	GaussianObject (Ours)	4.98	24.81	0.9350	3.63	27.00	0.9512	2.75	28.62	0.9638
	CF-GaussianObject (Ours)	8.47	21.39	0.9014	5.71	24.06	0.9269	5.50	24.39	0.9300
OmniObject3D	DVGO [Sun et al. 2022]	14.48	17.14	0.8952	12.89	18.32	0.9142	11.49	19.26	0.9302
	3DGS [Kerbl et al. 2023]	8.60	17.29	0.9299	7.74	18.29	0.9378	6.50	20.26	0.9483
	DietNeRF [Jain et al. 2021]	11.64	18.56	0.9205	10.39	19.07	0.9267	10.32	19.26	0.9258
	RegNeRF [Niemeyer et al. 2022]	16.75	15.20	0.9091	14.38	15.80	0.9207	10.17	17.93	0.9420
	FreeNeRF [Yang et al. 2023]	8.28	17.78	0.9402	7.32	19.02	0.9464	7.25	20.35	0.9467
	SparseNeRF [Guangcong et al. 2023]	17.47	15.22	0.8921	21.71	15.86	0.8935	23.76	17.16	0.8947
	ZeroRF [Shi et al. 2024b]	4.44	27.78	0.9615	3.11	31.94	0.9731	3.10	32.93	0.9747
	FSGS [Zhu et al. 2024]	6.25	24.71	0.9545	6.05	26.36	0.9582	4.17	29.16	0.9695
	GaussianObject (Ours)	2.07	30.89	0.9756	1.55	33.31	0.9821	1.20	35.49	0.9870
	CF-GaussianObject (Ours)	2.62	28.51	0.9669	2.03	30.73	0.9738	2.08	31.23	0.9757

Table 3. Quantitative comparisons on the OpenIllumination dataset. Methods with † means the metrics are from the ZeroRF paper [Shi et al. 2024b].

Method	4-view			6-view		
	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑
DVGO	11.84	21.15	0.8973	8.83	23.79	0.9209
3DGS	30.08	11.50	0.8454	29.65	11.98	0.8277
DietNeRF†	10.66	23.09	0.9361	9.51	24.20	0.9401
RegNeRF†	47.31	11.61	0.6940	30.28	14.08	0.8586
FreeNeRF†	35.81	12.21	0.7969	35.15	11.47	0.8128
SparseNeRF	22.28	13.60	0.8808	26.30	12.80	0.8403
ZeroRF†	9.74	24.54	0.9308	7.96	26.51	0.9415
Ours	6.71	24.64	0.9354	5.44	26.54	0.9443

Fig. 6. Qualitative results on the OpenIllumination dataset. Although ZeroRF shows competitive PSNR and SSIM, its renderings often appear blurred. While GaussianObject outperforms in restoring fine details, achieving a significant perceptual quality advantage.

Performance of CF-GaussianObject. CF-GaussianObject is evaluated on the MipNeRF360 and OmniObject3D datasets, with results detailed in Table 2 and Fig. 5. Though CF-GaussianObject exhibits some performance degradation, it eliminates the need for precise camera parameters, significantly enhancing its practical utility. Its performance remains competitive compared to other SOTA methods that depend on accurate camera parameters. Notably, we observe that the performance degradation correlates with an increase in the number of input views, primarily due to declines in the accuracy of DUSt3R's estimates as the number of views rises. As demonstrated in Fig. 7, comparative experiments on smartphone-captured images confirm the superior reconstruction capabilities and visual quality of CF-GaussianObject. More visualization of CF-GaussianObject can be found in our appendix and supplementary materials.

4.4 Ablation Studies

Key Components. We conduct a series of experiments to validate the effectiveness of each component. The following experiments are performed on MipNeRF360 with 4 input views, and averaged metric values are reported. We disable visual hull initialization, floater elimination, Gaussian repair model setup, and Gaussian repair process once at a time to verify their effectiveness. The Gaussian repair loss is further compared with the Score Distillation Sampling (SDS) loss [Poole et al. 2023], and the depth loss is ablated. The results, presented in Table 5 and Fig. 9, indicate that each element significantly contributes to performance, with their absence leading to a decline in results. Particularly, omitting visual hull initialization results in a marked decrease in performance. Gaussian repair model setup and the Gaussian repair process significantly enhance visual

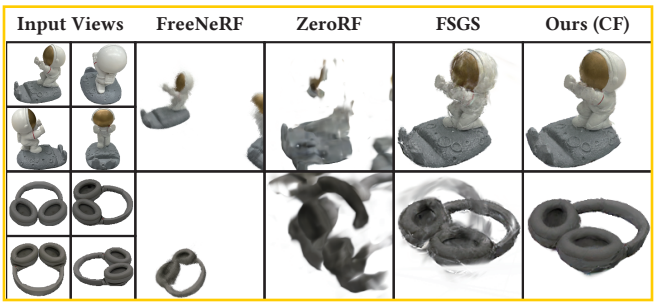

Fig. 7. Qualitative results on our-collected images captured by an iPhone 13. We equip other SOTAs with camera parameters predicted by DUSt3R for fair comparison. The results demonstrate the superior performance of our CF-GaussianObject among casually captured images, with fine details and higher visual quality.

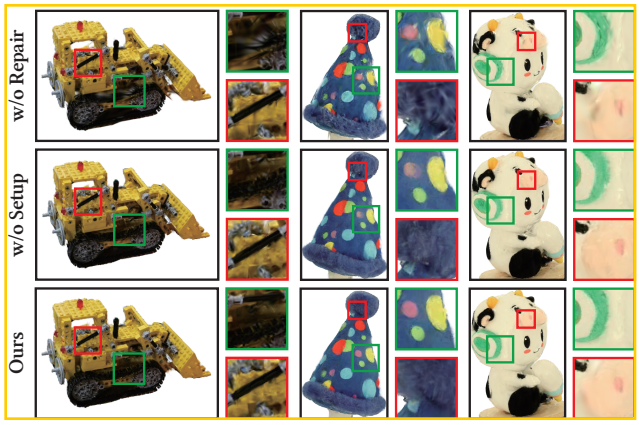

Fig. 8. Importance of our Gaussian repair model setup. Without the Gaussian repair process or the finetuning of the ControlNet, the renderings exhibit noticeable artifacts and lack of details, particularly in areas with insufficient view coverage. Zoom in for better comparison.

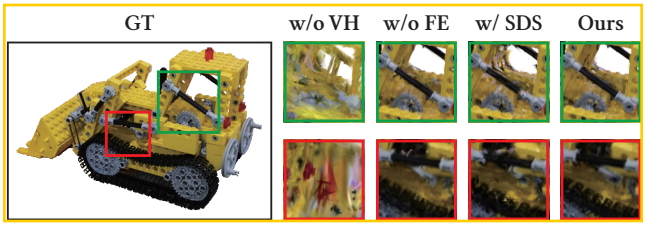

Fig. 9. Ablation study on different components. "VH" denotes for visual hull and "FE" is floater elimination. The "GT" image is from a test view.

quality, and the absence of either results in a substantial decline in perceptual quality as shown in Fig. 8. Contrary to its effectiveness in text-to-3D or single image-to-3D, SDS results in unstable optimization and diminished performance in our context. The depth loss shows marginal promotion, mainly for LPIPS and SSIM. We apply it to enhance the robustness of our framework.

Structure of Repair Model. Our repair model is designed to generate photo-realistic and 3D-consistent views of the target object. This

Table 4. Quantitative comparisons with LRM-like methods on MipNeRF360.

Method	LPIPS* ↓	PSNR ↑	SSIM ↑
TGS [Zou et al. 2024]	9.14	18.07	0.9073
LGM-4 [Tang et al. 2024a]	9.20	17.97	0.9071
LGM-1 [Tang et al. 2024a]	9.13	17.46	0.9071
GaussianObject (Ours)	**4.99**	**24.81**	**0.9350**

Fig. 10. Qualitative comparisons by ablating different Gaussian repair model setup methods. "MDepth" denotes the repair model with masked monocular depth estimation as the condition.

Table 5. Ablation study on key components.

Method	LPIPS* ↓	PSNR ↑	SSIM ↑
Ours w/o Visual Hull	12.72	15.95	0.8719
Ours w/o Floater Elimination	4.99	24.73	0.9346
Ours w/o Setup	5.53	24.28	0.9307
Ours w/o Gaussian Repair	5.55	24.37	0.9297
Ours w/o Depth Loss	5.09	**24.84**	0.9341
Ours w/ SDS [Poole et al. 2023]	6.07	22.42	0.9188
GaussianObject (Ours)	**4.98**	24.81	**0.9350**

is achieved by leave-one-out training and perturbing the attributes of 3D Gaussians to create image pairs for fine-tuning a pre-trained image-conditioned ControlNet. Similarities can be found in Dreambooth [Ruiz et al. 2023], which aims to generate specific subject images from limited inputs. To validate the efficacy of our design, we evaluate the samples generated by our Gaussian repair model and other alternative structures. The first is implemented with Dreambooth [Raj et al. 2023; Ruiz et al. 2023], which embeds target object priors with semantic modifications. To make the output corresponding to the target object, we utilize SDEdit [Meng et al. 2022] to guide the image generation. Inspired by Song et al. [2023b], the second introduces a monocular depth conditioning ControlNet [Zhang et al. 2023a], which is fine-tuned using data pair generation as in Sec. 3.4. We also assess the performance using masked depth conditioning. Furthermore, we consider Zero123-XL [Deitke et al. 2023; Liu et al. 2023c], a well-known single-image reconstruction model requiring object-centered input images with precise camera rotations and positions. Here, we manually align the coordinate system and select the closest image to the novel viewpoint as its reference.

The results, as shown in Table 6 and Fig. 10, reveal that semantic modifications proposed by Dreambooth alone fail in 3D-coherent synthesis. Monocular depth conditioning, whether with or without masks, despite some improvements, still struggles with depth

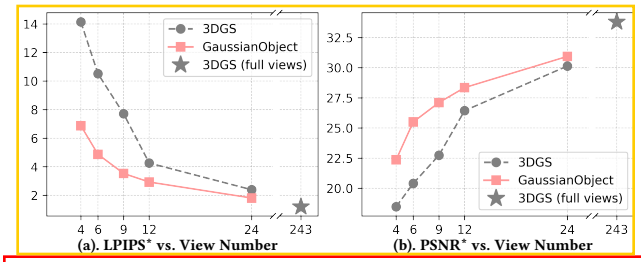

Fig. 11. Ablation on Training View Number. Experiments are conducted on scene *kitchen* in the MipNeRF360 dataset.

Table 6. Ablation study about alternatives of the Gaussian repair model.

Method	LPIPS* ↓	PSNR ↑	SSIM ↑
Zero123-XL [Liu et al. 2023c]	13.97	17.71	0.8921
Dreambooth [Ruiz et al. 2023]	6.58	21.85	0.9093
Depth Condition	7.00	21.87	0.9112
Depth Condition w/ Mask	6.87	21.92	0.9117
GaussianObject (Ours)	**5.79**	**23.55**	**0.9220**

roughness and artifacts. Zero123-XL, while generating visually acceptable images, the multi-view structure consistency is lacking. In contrast, our model excels in both 3D consistency and detail fidelity, outperforming others qualitatively and quantitatively.

Effect of View Numbers. We design experiments to evaluate the advantage of our method over different training views. As shown in Fig. 11, GaussianObject consistently outperforms vanilla 3DGS in varying numbers of training views. Besides, GaussianObject with 24 training views achieves performance comparable to that of 3DGS trained on all views (243).

4.5 Limitations and Future Work

GaussianObject demonstrates notable performance in sparse 360° object reconstruction, yet several avenues for future research exist. In regions completely unobserved or insufficiently observed, our repair model may generate hallucinations, *i.e.*, it may produce non-existent details, as shown in Fig. 12. However, these regions are inherently non-deterministic in information, and other methods also struggle in these areas. Additionally, due to the high sparsity level, our model is currently limited in capturing view-dependent effects. With such sparse data, our method cannot differentiate whether the appearance is view-dependent or inherent. Consequently, it 'bakes in' the view-dependent features (like reflected white light) onto the surface, resulting in an inability to display view-dependent appearance from novel viewpoints correctly and leading to some unintended artifacts as demonstrated in Fig. 13. Fine-tuning diffusion models with more view-dependent data may be a promising direction. Besides, integrating GaussianObject with surface reconstruction methods like 2DGS [Huang et al. 2024] and GOF [Yu et al. 2024] is a promising direction. Furthermore, CF-GaussianObject achieves competitive performance, but there is still a performance gap compared to precise camera parameters. An interesting exploration is to leverage confidence maps from matching methods for more accurate pose estimation.

Fig. 12. Hallucinations of non-existent details. GaussianObject may fabricate visually reasonable details in areas with little information. For instance, the hole in the stone vase is filled in.

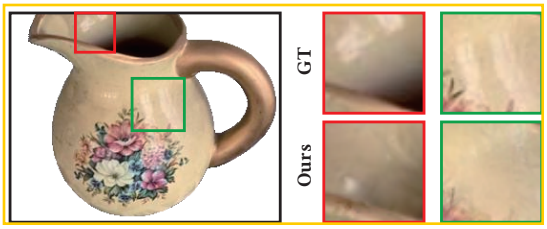

Fig. 13. Comparative visualization highlighting the challenge of reconstructing view-dependent appearance with only four input images.

5 CONCLUSION

In summary, GaussianObject is a novel framework designed for high-quality 3D object reconstruction from extremely sparse 360° views, based on 3DGS with real-time rendering capabilities. We design two main methods to achieve this goal: structure-prior-aided optimization for facilitating the multi-view consistency construction and a Gaussian repair model to remove artifacts caused by omitted or highly compressed object information. We also provide a COLMAP-free version that can be easily applied in real life with competitive performance. We sincerely hope that GaussianObject can advance daily-life applications of reconstructing 3D objects, markedly reducing capture requirements and broadening application prospects.

ACKNOWLEDGMENTS

This work was supported by the NSFC under Grant 62322604 and 62176159, and in part by the Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102. The authors express gratitude to the anonymous reviewers for their valuable feedback and to Deyu Wang for his assistance with figure drawing and Blender support.

REFERENCES

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2021. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 5460–5469.

Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. 2023. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288* (2023).

James Burgess, Kuan-Chieh Wang, and Serena Yeung. 2024. Viewpoint Textual Inversion: Unleashing Novel View Synthesis with Pretrained 2D Diffusion Models. *ECCV* (2024).

Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Chen Yang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, and Qi Tian. 2023. Segment Anything in 3D with NeRFs. In *NeurIPS*.

Eric R Chan, Koki Nagano, Matthew A Chan, Alexander W Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. 2023.

GeNVS: Generative novel view synthesis with 3D-aware diffusion models.

Jonathan Chang. 2023. minLoRA. https://github.com/cccntu/minLoRA.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. 2024. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*. 19457–19467.

Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. 2023. Fantasia3D: Disentangling Geometry and Appearance for High-quality Text-to-3D Content Creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 22246–22256.

Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. 2023. Depth-regularized optimization for 3d gaussian splatting in few-shot images. *arXiv preprint arXiv:2311.13398* (2023).

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. 2023. Objaverse-XL: A Universe of 10M+ 3D Objects. *arXiv preprint arXiv:2307.05663* (2023).

Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. 2022. Depth-supervised NeRF: Fewer Views and Faster Training for Free. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12872–12881. https://doi.org/10.1109/CVPR52688.2022.01254

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. https://openreview.net/forum?id=YicbFdNTTy

Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, Zhangyang Wang, and Yue Wang. 2024. InstantSplat: Unbounded Sparse-view Pose-free Gaussian Splatting in 40 Seconds. arXiv:2403.20309 [cs.CV].

Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A Efros, and Xiaolong Wang. 2024. Colmap-free 3d gaussian splatting. *CVPR* (2024).

Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. 2024. CAT3D: Create Anything in 3D with Multi-View Diffusion Models. *arXiv preprint arXiv:2405.10314* (2024).

Guangcong Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. 2023. SparseNeRF: Distilling Depth Ranking for Few-shot Novel View Synthesis. *IEEE/CVF International Conference on Computer Vision (ICCV)* (2023).

Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. 2023. threestudio: A unified framework for 3D content generation. https://github.com/threestudio-project/threestudio.

Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. 2023. Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. 2024. Lrm: Large reconstruction model for single image to 3d. *ICLR* (2024).

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. *ICLR* (2022).

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery. https://doi.org/10.1145/3641519.3657428

Shruti Jadon. 2020. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. 1–7. https://doi.org/10.1109/CIBCB48159.2020.9277638

Ajay Jain, Matthew Tancik, and Pieter Abbeel. 2021. Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 5865–5874. https://doi.org/10.1109/ICCV48922.2021.00583

Wonbong Jang and Lourdes Agapito. 2024. NViST: In the Wild New View Synthesis from a Single Image with Transformers. *CVPR* (2024).

Hanwen Jiang, Zhenyu Jiang, Yue Zhao, and Qixing Huang. 2024. LEAP: Liberate Sparse-view 3D Modeling from Camera Poses. *ICLR* (2024).

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (2023).

Mijeong Kim, Seonguk Seo, and Bohyung Han. 2022. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *CVPR*. 12912–12921.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6114

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. *ICCV* (2023).

A. Laurentini. 1994. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16, 2 (1994), 150–162. https://doi.org/10.1109/34.273735

Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. 2024. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. *ICLR* (2024).

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. 2023. Magic3D: High-Resolution Text-to-3D Content Creation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Isabella Liu, Linghao Chen, Ziyang Fu, Liwen Wu, Haian Jin, Zhong Li, Chin Ming Ryan Wong, Yi Xu, Ravi Ramamoorthi, Zexiang Xu, and Hao Su. 2023a. OpenIllumination: A Multi-Illumination Dataset for Inverse Rendering Evaluation on Real Objects. *NeuRIPS 2023*.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. 2023c. Zero-1-to-3: Zero-shot One Image to 3D Object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 9298–9309.

Xinhang Liu, Shiu-hong Kao, Jiaben Chen, Yu-Wing Tai, and Chi-Keung Tang. 2023b. Deceptive-NeRF: Enhancing NeRF Reconstruction using Pseudo-Observations from Diffusion Models. *arXiv preprint arXiv:2305.15171* (2023).

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Bkg6RiCqY7.

Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. 2024. Gaussian Splatting SLAM. In *CVPR*.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. 2022. Sdedit: Guided image synthesis and editing with stochastic differential equations. *ICLR* (2022).

Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. 2023. Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12663–12673. https://doi.org/10.1109/CVPR52729.2023.01218.

Norman Müller, Katja Schwarz, Barbara Rössle, Lorenzo Porzi, Samuel Rota Bulò, Matthias Nießner, and Peter Kontschieder. 2024. MultiDiff: Consistent Novel View Synthesis from a Single Image. In *CVPR*. 10258–10268.

Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. 2022. RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5470–5480. https://doi.org/10.1109/CVPR52688.2022.00540

Avinash Paliwal, Wei Ye, Jinhui Xiong, Dmytro Kotovenko, Rakesh Ranjan, Vikas Chandra, and Nima Khademi Kalantari. 2024. CoherentGS: Sparse Novel View Synthesis with Coherent 3D Gaussians. *ECCV* (2024).

Zijie Pan, Zeyu Yang, Xiatian Zhu, and Li Zhang. 2024. Fast Dynamic 3D Object Generation from a Single-view Video. *arXiv preprint arXiv:2401.08742* (2024).

Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. 2021. HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.* 40, 6 (2021), 238:1–238:12.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. 2023. Dreamfusion: Text-to-3d using 2d diffusion. *ICLR* (2023).

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 8748–8763. https://proceedings.mlr.press/v139/radford21a.html

Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. 2023. Dreambooth3d: Subject-driven text-to-3d generation. *ICCV* (2023).

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. 2021. Vision Transformers for Dense Prediction. *ICCV* (2021).

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. 2020. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence* 44, 3 (2020), 1623–1637.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. 2022. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 3 (2022).

Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. 2022. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*. 12892–12901.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10674–10685. https://doi.org/10.1109/CVPR52688.2022.01042

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*. 22500–22510.

Johannes Lutz Schonberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Seunghyeon Seo, Yeonjin Chang, and Nojun Kwak. 2023. Flipnerf: Flipped reflection rays for few-shot novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 22883–22893.

Ruizhi Shao, Jingxiang Sun, Cheng Peng, Zerong Zheng, Boyao Zhou, Hongwen Zhang, and Yebin Liu. 2024. Control4D: Efficient 4D Portrait Editing with Text. *CVPR* (2024).

Ruoxi Shi, Xinyue Wei, Cheng Wang, and Hao Su. 2024b. ZeroRF: Fast Sparse View 360° Reconstruction with Zero Pretraining. *CVPR* (2024).

Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. 2024a. MV-Dream: Multi-view Diffusion for 3D Generation. *ICLR* (2024).

Nagabhushan Somraj, Adithyan Karanayil, Sai Harsha Mupparaju, and Rajiv Soundararajan. 2024. Simple-RF: Regularizing Sparse Input Radiance Fields with Simpler Solutions. *arXiv preprint arXiv:2404.19015* (2024).

Nagabhushan Somraj, Adithyan Karanayil, and Rajiv Soundararajan. 2023. SimpleNeRF: Regularizing Sparse Input Neural Radiance Fields with Simpler Solutions. In *SIGGRAPH Asia 2023 Conference Papers (SA '23)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3610548.3618188

Nagabhushan Somraj and Rajiv Soundararajan. 2023. ViP-NeRF: Visibility Prior for Sparse Input Neural Radiance Fields. (August 2023). https://doi.org/10.1145/3588432.3591539

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. *ICLR* (2021).

Jiuhn Song, Seonghoon Park, Honggyu An, Seokju Cho, Min-Seop Kwak, Sungjin Cho, and Seungryong Kim. 2023b. DaRF: Boosting Radiance Fields from Sparse Inputs with Monocular Depth Adaptation. *2023 NIPS* (2023).

Liangchen Song, Zhong Li, Xuan Gong, Lele Chen, Zhang Chen, Yi Xu, and Junsong Yuan. 2023a. Harnessing Low-Frequency Neural Fields for Few-Shot View Synthesis. *arXiv preprint arXiv:2303.08370* (2023).

Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *CVPR*.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. 2024a. LGM: Large Multi-View Gaussian Model for High-Resolution 3D Content Creation. *ECCV* (2024).

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. 2024b. DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation. *ICLR* (2024).

Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. 2023a. Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12619–12629. https://doi.org/10.1109/CVPR52729.2023.01214

Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. 2024b. PF-LRM: Pose-Free Large Reconstruction Model for Joint Pose and Shape Prediction. *ICLR* (2024).

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. 2024a. DUSt3R: Geometric 3D Vision Made Easy. *CVPR* (2024).

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. 2023b. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. 2021. NeRF–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064* (2021).

Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. 2024. MeshLRM: Large Reconstruction Model for High-Quality Mesh. *arXiv preprint arXiv:2404.12385* (2024).

Zhenzhen Weng, Jingyuan Liu, Hao Tan, Zhan Xu, Yang Zhou, Serena Yeung-Levy, and Jimei Yang. 2023. Template-Free Single-View 3D Human Digitalization with Diffusion-Guided LRM. *Preprint* (2023).

Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P. Srinivasan, Dor Verbin, Jonathan T. Barron, Ben Poole, et al. 2024. ReconFusion: 3D Reconstruction with Diffusion Priors. *CVPR* (2024).

Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, Dahua Lin, and Ziwei Liu. 2023. OmniObject3D: Large-Vocabulary 3D Object Dataset for Realistic Perception, Reconstruction and Generation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), 803–814. https://api.semanticscholar.org/CorpusID:255998491

Jamie Wynn and Daniyar Turmukhambetov. 2023. DiffusioNeRF: Regularizing Neural Radiance Fields with Denoising Diffusion Models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4180–4189. https://doi.org/10.1109/CVPR52729.2023.00407

Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. 2023. SparseGS: Real-Time 360° Sparse View Synthesis using Gaussian Splatting. *Arxiv* (2023).

Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. 2022. SinNeRF: Training Neural Radiance Fields on Complex Scenes from a Single Image. In *Computer Vision – ECCV 2022 (Lecture Notes in Computer Science)*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 736–753. https://doi.org/10.1007/978-3-031-20047-2_42

Dejia Xu, Ye Yuan, Morteza Mardani, Sifei Liu, Jiaming Song, Zhangyang Wang, and Arash Vahdat. 2024c. AGG: Amortized Generative 3D Gaussians for Single Image to 3D. *arXiv preprint 2401.04099* (2024).

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

Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. 2024b. DMV3D: Denoising Multi-View Diffusion using 3D Large Reconstruction Model. *ICLR* (2024).

Jiawei Yang, Marco Pavone, and Yue Wang. 2023. FreeNeRF: Improving Few-Shot Neural Rendering with Free Frequency Regularization. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8254–8263. https://doi.org/10.1109/CVPR52729.2023.00798

Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. GaussianDreamer: Fast Generation from Text to 3D Gaussians by Bridging 2D and 3D Diffusion Models. *CVPR* (2024).

Zehao Yu, Torsten Sattler, and Andreas Geiger. 2024. Gaussian Opacity Fields: Efficient High-quality Compact Surface Reconstruction in Unbounded Scenes. *arXiv:2404.10772* (2024).

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. 2024. GS-LRM: Large Reconstruction Model for 3D Gaussian Splatting. *arXiv* (2024).

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023a. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3836–3847.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023b. ControlNet-v1-1-nightly. https://github.com/lllyasviel/ControlNet-v1-1-nightly

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

Zhizhuo Zhou and Shubham Tulsiani. 2023. SparseFusion: Distilling View-Conditioned Diffusion for 3D Reconstruction. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12588–12597. https://doi.org/10.1109/CVPR52729.2023.01211

Junzhe Zhu and Peiye Zhuang. 2024. HiFA: High-fidelity Text-to-3D Generation with Advanced Diffusion Guidance. *ICLR* (2024).

Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. 2024. FSGS: Real-Time Few-shot View Synthesis using Gaussian Splatting. *ECCV* (2024).

Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. 2024. Triplane Meets Gaussian Splatting: Fast and Generalizable Single-View 3D Reconstruction with Transformers. *CVPR* (2024).

A APPENDIX

The contents of this appendix include:

(1) Dataset Details (A.1).
(2) Implementation Details (A.2).
(3) Experiment Details of Comparison (A.3).
(4) More Experimental Results (A.4).
(5) Ablation Study Details (A.5).
(6) Supplementary Video Descriptions (A.6).

A.1 Dataset Details

To ensure a rigorous evaluation, we employ SAM-based methods [Cen et al. 2023; Kirillov et al. 2023] to generate consistent object masks of test views. While this process is necessary for benchmarking in our study, it is not required for practical applications. In real-world scenarios, the system necessitates only four masks from captured images, which are straightforward to obtain with any segmentation method.

MipNeRF360. The sparse MipNeRF360 dataset, derived from the dataset provided by Barron et al. [2021], focuses on three scenes containing a primary object: bonsai, garden, and kitchen. For performance evaluation, the scenes are tested using images downsampled by a factor of 4×, following the train-test splits from Wu et al. [2024].

OmniObject3D. OmniObject3D includes 6k real 3D objects in 190 large-vocabulary categories. We selected 17 objects from OmniObject3D [Wu et al. 2023]. The items chosen include: *back-pack_016*, *box_043*, *broccoli_003*, *corn_007*, *dinosaur_006*, *flower_pot_007*, *gloves_009*, *guitar_002*, *hamburger_012*, *picnic_basket_009*, *pineapple_013*, *sandwich_003*, *suitcase_006*, *timer_010*, *toy_plane_005*, *toy_truck_037*, and *vase_012* for a fair evaluation. We manually choose camera views for training and use every eighth view left for testing. Most scenes are originally in 1080p resolution, while *gloves_009* and *timer_010* are in 720p resolution. To maintain consistency across all scenes, we upscaled the images from these two scenes to 1080p resolution. All the images are downsampled to a factor of 2×.

OpenIllumination. OpenIllumination is a real-world dataset captured by the LightStage. We use the sparse OpenIllumination dataset proposed in ZeroRF [Shi et al. 2024b]. Note that ZeroRF re-scaled and re-centered the camera poses to align the target object with the world center. We test on the sparse OpenIllumination dataset [Liu et al. 2023a] with provided object masks, which are generated by SAM and train-test splits, downscaling the images by a factor of 4×, following the same experimental setting as in Shi et al. [2024b].

Our-collected Unposed Images. To better align with daily usage, we captured four images of common objects using an iPhone 13 from approximately front, back, left, and right directions without strict requirements on camera positioning or angles. We employed DUSt3R [Wang et al. 2024a] to predict the camera parameters, including intrinsics and poses, for these images. Additionally, we used SAM [Kirillov et al. 2023] along with the iPhone's native segmentation features to perform image segmentation. All comparative methods also included these camera parameters and masks.

A.2 Implementation Details

We build our GaussianObject upon 3DGS [Kerbl et al. 2023] and Threestudio [Guo et al. 2023]. For visual hull reconstruction, we adopt a coarse-to-fine approach, starting with rough spatial sampling to estimate the bounding box and then proceeding to detailed sampling within it. Since the sparse input views cannot provide sufficient multi-view consistency, we reconstruct all objects with only two degrees of spherical harmonics (SH). We adopt the densification and opacity reset methods from vanilla 3DGS, conducting densification every 100 iterations and resetting opacity every 1,000 iterations. As for floater elimination, we start it from 500 iterations and conduct it every 500 iterations until 6k iterations. The adaptive threshold τ is initially set to the mean plus the standard deviation of the average distance calculated via KNN and is linearly decreased to 0 in 6k iterations. The loss weight for $\mathcal{L}_{\text{gs}}$ is set to: $\lambda_{\text{SSIM}} = 0.2$, $\lambda_{\text{m}} = 0.001$ and $\mathcal{L}_{\text{d}} = 0.0005$.

For the Gaussian repair model setup, we use leave-one-out training to generate image pairs and 3D Noise ϵ_s. We use the visual hull corresponding to N reference images to initialize all 3DGS models during leave-one-out training. We add noises to all the attributes of 3D Gaussians except for SH coefficients. Whenever we need a data pair from adding 3D noises, we use the mean μ_Δ and variance σ_Δ to generate new noisy Gaussians for rendering, thus enabling sufficient data generation. All the images are constantly padded or center-cropped before being fed into the Gaussian repair model to preserve the real ratio of the target object. At the first iteration of training the Gaussian repair model, manual noise generated according to distribution is used for training with a 100% probability. Each time training is conducted with manual noise, this probability is reduced by 0.5%, to utilize cached images from leave-one-out training increasingly, as illustrated in Algorithm 1. Referred to Dreambooth [Ruiz et al. 2023], the [V] used for object-specific text prompt is "xxy5syt00". We apply LoRA [Hu et al. 2022] in the fine-tuning process. LoRA optimizes a learnable low-rank residual matrix $\Delta W_i = A_i B_i$, where $A_i \in \mathbb{R}^{m\times r}$, $B_i \in \mathbb{R}^{r\times n}$. The hyperparameter of the LoRA rank $r \ll m, n$. We add LoRA layers to all embedding, linear, and convolution layers of the transformer blocks in the diffusion U-Net, the ControlNet U-Net, and the CLIP model. LoRA rank r is set to 64, and the learning rate is set to 10^{-3}. We fine-tune the Gaussian repair model for 1800 steps, optimized using an AdamW optimizer [Loshchilov and Hutter 2019] with β values of $(0.9, 0.999)$.

For the Gaussian repair process, we optimize the coarse 3DGS model for 4k steps, involving both a reference image and a repaired novel view image for the first 2800 steps supervised by $\mathcal{L}_{\text{rep}}$ and $\mathcal{L}_{\text{gs}}$ together, with the weight of $\mathcal{L}_{\text{rep}}$ progressively decayed from 1.0 to 0.1. The final 1200-step training only involves reference views. Based on the camera poses of N reference views, we estimate an elliptic trajectory that encircles the object and find N points on the trajectory closest to the reference views, respectively. These N points divide the ellipse into N segments of the arc. On each arc segment, novel views are only sampled from the middle 80% of the arc (repair path). The remaining arcs constitute reference paths. To expedite training, we avoid using the time-consuming DDIM process [Song et al. 2021] for every novel view rendering. Instead,

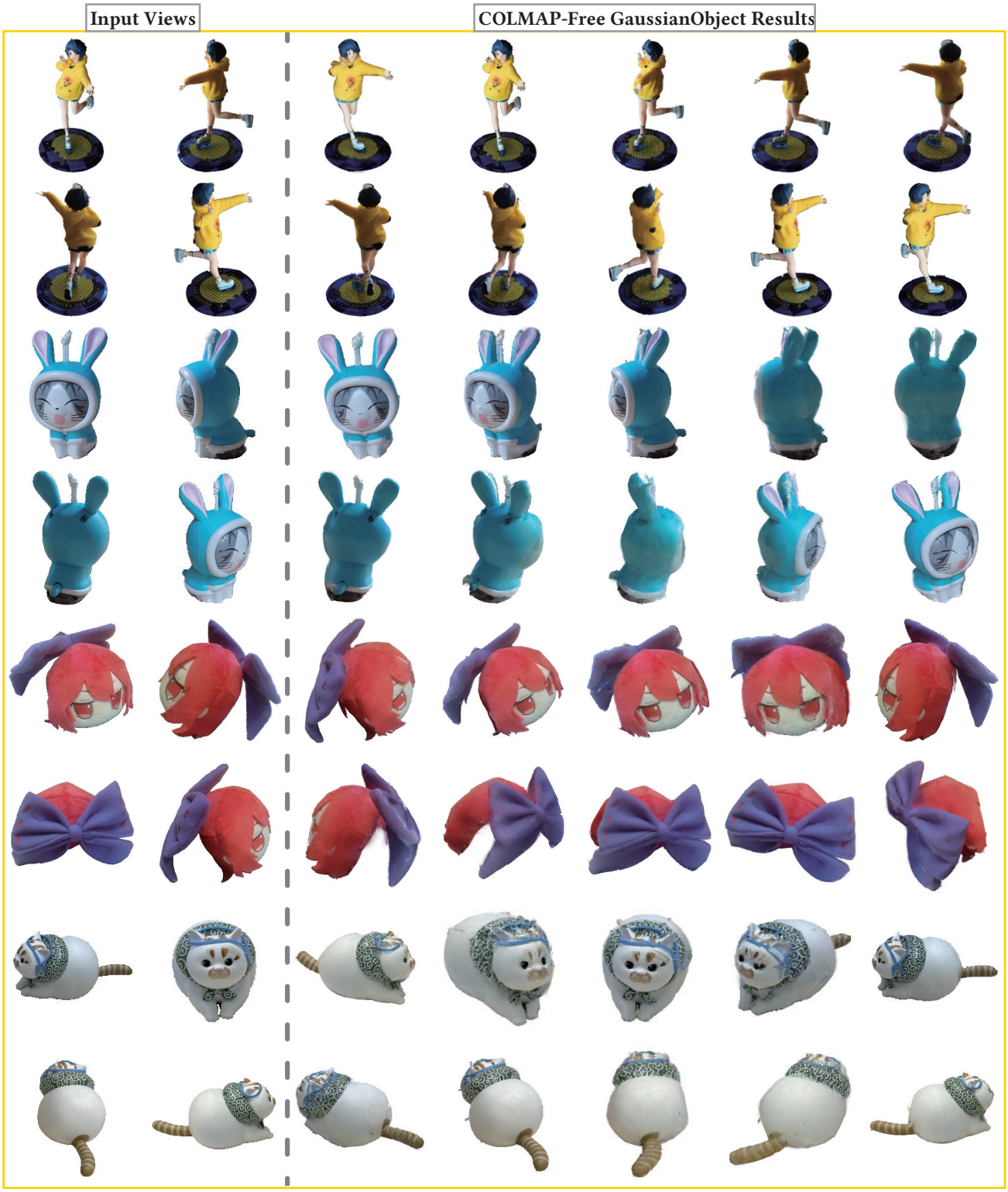

Fig. 14. Performance of CF-GaussianObject among iPhone captured images. By introducing modified DUSt3R into our GaussianObject, we successfully achieve COLMAP-free reconstruction from extremely sparse views.

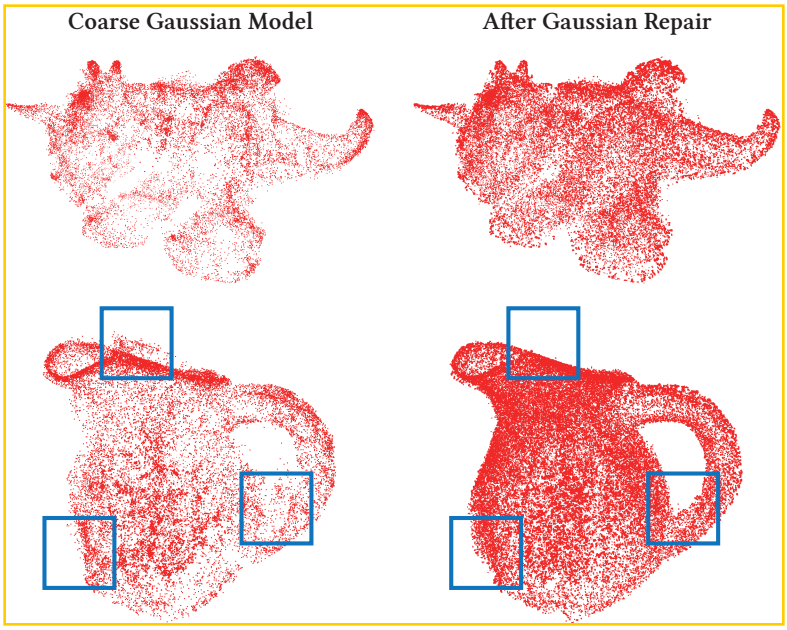

Fig. 15. Point clouds of GaussianObject before and after Gaussian repair. In the *dinosaur* scene, the Gaussian repair model significantly enhances the point cloud density, resulting in a more distinct and clear representation of the dinosaur's body. In the *vase* scene, noticeable artifacts around the handle and the mouth are effectively eliminated by the Gaussian repair model, and the incomplete bottom of the vase is repaired. These key areas are highlighted with blue boxes.

Algorithm 1 Gaussian Repair Model Data Generation Algorithm

Input: Gaussian Repair Model $\mathcal{R}$, Coarse 3DGS Model $\mathcal{G}_c$, Mean $\mu_{\Delta,a}$ and Variance $\sigma_{\Delta,a}$ for Each Attribute a of $\mathcal{G}_c$, N Leave-one-out 3DGS Models $\{\hat{\mathcal{G}}_c^i\}_{i=0}^{N-1}$

$P_{\text{manual}} \leftarrow 1$
Sample p from $U[0,1]$
for each iteration **do**
  Sample an index i from $\{0, 1, \ldots, N-1\}$
  if $p < P_{\text{manual}}$ **then**
    $\mathcal{G}_c(\epsilon_s) \leftarrow \mathcal{G}_c$
    for all attribute a of $\mathcal{G}_c$ except for SH coefficients **do**
      Sample $\epsilon_{s,a}$ from $N(\mu_{\Delta,a}, \sigma_{\Delta,a}^2)$
      $a \leftarrow a + \epsilon_{s,a}$
    end for
    $x' \leftarrow x'(\mathcal{G}_c(\epsilon_s), \pi^i)$
    $P_{\text{manual}} \leftarrow 0.995 \times P_{\text{manual}}$
  else
    $x' \leftarrow x'(\hat{\mathcal{G}}_c^i, \pi_i)$
  end if
  Optimize $\mathcal{R}$ with data pair (x', x_i)
end for

we employ cached images for optimization. Specifically, at the beginning of every 200 iterations, we sample and repair two novel views from each repair path. Throughout these 200 iterations, we utilize these images to repair 3D Gaussians. We set the weights for $\mathcal{L}_{\text{rep}}$ as follows: $\lambda_1 = 0.5$, $\lambda_2 = 0.5$ and $\lambda_p = 2.0$ accross all experiments. We employ densification and opacity resetting to regulate the number of Gaussians. The Gaussian densification process is effective from 400 to 3600 steps. The Gaussian model undergoes densification and pruning at intervals of every 100 steps, and its opacity is reset every 500 steps.

For CF-GaussianObject, we modify the DUSt3R [Wang et al. 2024a] pipeline to better suit our task. We assume that all cameras share the same intrinsic, while the original DUSt3R predicts different camera focal lengths for each view. We initially use the average camera focal estimated from input views and further optimize it using DUSt3R's global optimization process. Background points are removed from DUSt3R's point cloud $\mathcal{P}$ using object masks and the predicted camera poses via a back-projection algorithm. The resultant $\mathcal{P}$ is much sparser than the visual hull point cloud, missing points on unseen surfaces and object fillers. we augment $\mathcal{P}$ with 10% of the points from the visual hull point cloud to densify the initial 3D Gaussian points. During the first 4000 steps of the 3DGS initial optimization process, we employ the tracking losses from MonoGS [Matsuki et al. 2024] and InstantSplat [Fan et al. 2024] to further optimize the camera poses $\hat{\Pi}^{\text{ref}}$ predicted by DUSt3R.

$$\hat{\pi}^{\text{ref}*} = \arg\min_{\hat{\pi}^{\text{ref}}} \|\mathcal{G}_c(\hat{\pi}^{\text{ref}}) - x^{\text{ref}}\|_1 + \lambda_{\text{pose}} \|\hat{\pi}^{\text{ref}} - \hat{\pi}_0^{\text{ref}}\|_1, \quad (12)$$

where $\hat{\pi}^{\text{ref}*}$ denotes the refined camera pose, and $\hat{\pi}_0^{\text{ref}}$ represents the initial camera pose from DUSt3R. λ_{pose} is introduced to ensure the refined poses do not deviate excessively from the initial ones, which

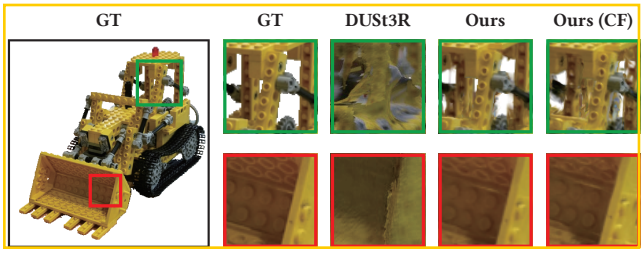

Fig. 16. Qualitative comparisons between DUSt3R and CF-GaussianObject with four input views.

is set to 0.0005 in our experiments. To evaluate CF-GaussianObject, we align the SfM camera poses with the noisy ones and optimize each test camera for up to 400 steps with the same loss during test time.

A.3 Experiment Details of Comparison

FSGS [Zhu et al. 2024] requires SfM points generated from the input images, which are too sparse in our setting of 4 input views. Alternatively, we randomly pick N_{pick} points from the SfM points with full images as input:

$$N_{\text{pick}} = \max\left(150, \frac{k_{\text{sparse}}}{k_{\text{all}}} N_{\text{all}}\right), \tag{13}$$

where k_{sparse} is the number of input images, k_{all} is the total number of images, and N_{all} is the total number of points.

A.4 More Experimental Results

A.4.1 Per-scene Performance. We provide per-scene qualitative metrics of MipNeRF360 in Tab. 10, Tab. 11 and Tab. 12; of OmniObject3D in Tab. 13, Tab. 14, Tab. 15, Tab. 16, Tab. 17 and Tab. 18; of OpenIllumination in Tab. 19 and Tab. 20. Additional qualitative comparisons on the OpenIllumination dataset are shown in Fig. 19.

A.4.2 Performance of CF-GaussianObject. Our COLMAP-free CF-GaussianObject bypasses the traditional Structure-from-Motion (SfM) pipeline requirements, enabling application to casually captured images. Utilizing an iPhone 13, we capture four images of common objects and obtain their corresponding masks using SAM [Kirillov et al. 2023]. We then employ CF-GaussianObject for the reconstruction of these images, with the results of the novel view synthesis displayed in Fig. 14. CF-GaussianObject delivers high-quality reconstructions from just four images without the need for precise camera parameters, producing images with exceptional visual quality and detailed richness. These results underscore CF-GaussianObject's effectiveness in achieving high-fidelity reconstruction from sparsely captured real-world images, showcasing its substantial potential for practical applications.

A.4.3 Comparison with DUSt3R. To further show the effectiveness of our design, we present qualitative comparisons between CF-GaussianObject and DUSt3R [Wang et al. 2024a] in Fig. 16. These results confirm that the strong performance stems from our design. Overall, these findings highlight the generalizability of CF-GaussianObject and its potential for real-world applications.

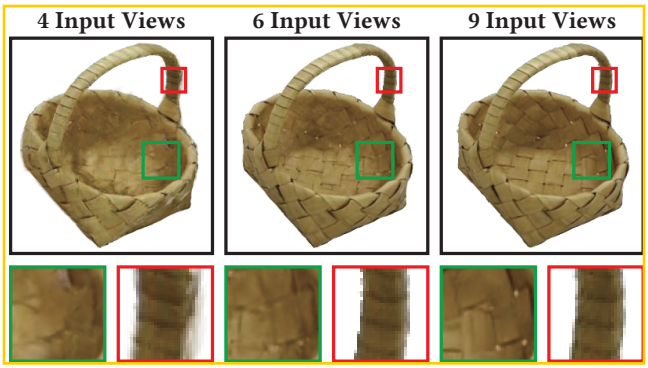

Fig. 17. Rendering performance with varying training views.

A.4.4 Influence of Gaussian repair process. We provide extra validations on our Gaussian repair process. Figure 15 illustrates the distribution of coarse 3D Gaussians and the refined distribution after the Gaussian repair. This comparison clearly demonstrates the significant enhancement in geometry achieved by our repair process. We also provide qualitative samples of the Gaussian repair model in Fig. 20, showing that the Gaussian repair model can effectively generate high-quality and consistent images from multiple views.

Table 7. Comparison of training time on one single RTX 3090 GPU.

Method	Time
DietNeRF [Jain et al. 2021]	8h+
RegNeRF [Niemeyer et al. 2022]	48h+
FreeNeRF [Yang et al. 2023]	24h+
SparseNeRF [Guangcong et al. 2023]	24h+
ZeroRF [Shi et al. 2024b]	~35min
ReconFusion [Wu et al. 2024]	8h+
GaussianObject	~30min
CF-GaussianObject (ours)	~33min

A.4.5 Training Time Comparison. We show the training time of GaussianObject and the various baselines on one single RTX 3090 GPU in Table 7. GaussianObject can be finished around 30 minutes, much faster than previous methods with higher quality. The process breakdown for GaussianObject is as follows: Initial optimization with structure priors takes approximately 1 minute; Gaussian repair model setup requires about 15 minutes; and Gaussian repair with distance-aware sampling lasts around 14 minutes.

For the COLMAP-free version, we incorporate a customized diff-gaussian-rasterization module during the coarse GS optimization. This module, which is not used in the standard GaussianObject nor the repair model setup and Gaussian repair, necessitates additional computations for gradient descent on camera poses and uses a faster rasterization module elsewhere. The training time for CF-GaussianObject typically totals about 33 minutes, broken down as follows: DUSt3R [Wang et al. 2024a] completes in roughly 17 seconds; initial optimization with structure priors takes about 3

Table 8. Robustness among randomness on MipNeRF360 dataset.

Performance	LPIPS* ↓	PSNR ↑	SSIM ↑
Mean	4.97	24.82	0.9345
Std	0.06	0.169	0.0007

minutes due to the customized diff-gaussian-rasterization module; Gaussian repair model setup remains at 15 minutes; and Gaussian repair with distance-aware sampling also takes about 14 minutes.

A.4.6 Robustness of Random Noise. Diffusion models rely on adding random noise followed by denoising to generate images; hence, their performance is heavily dependent on the random seed used. GaussianObject also employs a diffusion model to refine 3D Gaussian, so the diversity inherent in diffusion could affect the reconstruction quality of GaussianObject. To mitigate this diversity, we finetune the diffusion model and devise a distance-aware sampling strategy for Gaussian repair. To validate this, we conduct 10 independent experiments with different random seeds, and the results are presented in Table 8. Experimental results confirm the effectiveness of our design, demonstrating that randomness has little influence on our performance.

A.4.7 Performance on Reference and Repair Path. The rendering capabilities of GaussianObject are evaluated on both the reference and repair paths. To ensure the utmost fidelity to the reference views, the Gaussian repair process is exclusively applied along the repair path. Fig. 18 illustrates that GaussianObject delivers exceptional visual quality across both the reference and repair paths. Such results highlight the efficacy of our Gaussian repair process. GaussianObject consistently produces high fidelity and visual excellence renderings, irrespective of whether the views are from regions with sufficient or insufficient coverage.

A.4.8 Performance with Varying Views. We demonstrate the performance of GaussianObject under various training views in Fig. 17. As more views are provided, GaussianObject can reconstruct more details. This demonstrates the model's capacity to leverage additional perspectives for improved accuracy and detail in the reconstructed output. Specifically, with an increasing number of views, the visual quality of the renderings was significantly enhanced, reducing artifacts and producing finer details. This improvement highlights the robustness and scalability of GaussianObject in handling complex visual data.

A.4.9 Performance with view distribution. In this study, we evaluate the performance of GaussianObject on the 'kitchen' scene from the MipNeRF360 dataset across various view distributions. We systematically report the angles between input views in the 'view distribution' column of Table 9, with the first row reflecting the configuration used in the manuscript. Results reveal that GaussianObject consistently achieves robust performance across diverse view distributions, a capability that eludes many current Large Reconstruction Models (LRMs) [Tang et al. 2024a; Xu et al. 2024a; Zhang et al. 2024]. This robustness is crucial for practical applications where varied viewpoints are common.

Table 9. Performance in terms of view distribution.

View Distribution	LPIPS* ↓	PSNR ↑	SSIM ↑
110.4°, 112.4°, 65.1°, 72.1°	6.8716	22.36	0.9104
87.3°, 91.5°, 99.4°, 81.8°	6.8279	22.63	0.9098
60.4°, 164.2°, 51.1°, 84.3°	7.3637	21.72	0.9039
121.6°, 121.8°, 59.5°, 57.1°	6.9842	22.97	0.9094

A.5 Ablation Study Details

We employ ControlNet-Tile [Zhang et al. 2023a,b] as the Gaussian repair model of GaussianObject. ControlNet-Tile, based on Stable Diffusion v1.5, is designed to repair low-resolution or low-fidelity images by taking them as conditions. In our ablation study, we use other diffusion models as our Gaussian repair model in Sec.4.4. First, we use Stable Diffusion XL Image-to-Image [Meng et al. 2022], which cannot directly take images as input conditions. Therefore, we add 50% Gaussian noise to the novel view images and use the model to generate high-fidelity images by denoising the noisy images. Additionally, we evaluate ControlNet-ZoeDepth based on Stable Diffusion XL. ControlNet-ZoeDepth, which takes the ZoeDepth [Bhat et al. 2023] as a condition, is used to generate images aligned with the depth map. However, ZoeDepth may incorrectly assume that a single object with a white background is on a white table, leading to erroneous depth map predictions. In such cases, we test two settings: one using the entire predicted depth map and another using the object-masked depth map. Due to the ambiguity of monocular predicted depth, we add partial noise on the input image to preserve the image information, similar to SDEdit. Specifically, we add 50% Gaussian noise to the novel view images and set the weight scale of the ControlNet model to 0.55. In this experimental setup, we obtained the results of the ablation experiments on the structure of the Gaussian repair model.

A.6 Supplementary Video Descriptions

We prepare four sets of video comparisons for evaluation. The first set, *videos_1*, features videos rendered along trajectories encircling several objects, showcasing the performance of all tested models. These videos are organized by scene and the number of input views and are placed in different directories for ease of analysis.

In the second video set, *videos_2*, we provide more experiment videos. Each video is edited to demonstrate our results clearly. The video titled *refresh.mp4* compares all the tested models with GaussianObject on 9 scenes with 4 input views. Videos labeled with the suffix *_compare.mp4* focus on comparing individual methods with GaussianObject. The video titled *transfer_3DGS_to_GaussianObject.mp4* sequentially illustrates the enhancements we have implemented on the 3DGS, detailing each step of our improvement process. Additionally, the video *COLMAP_free.mp4* shows our COLMAP-free experiments using our own collected unposed images, presenting both the input images and reconstruction results, along with comparisons to other methods.

In the third video set, *videos_3*, we present videos from COLMAP-free experiments. Each video is labeled according to the dataset, scene, and number of input views involved in the experiment.

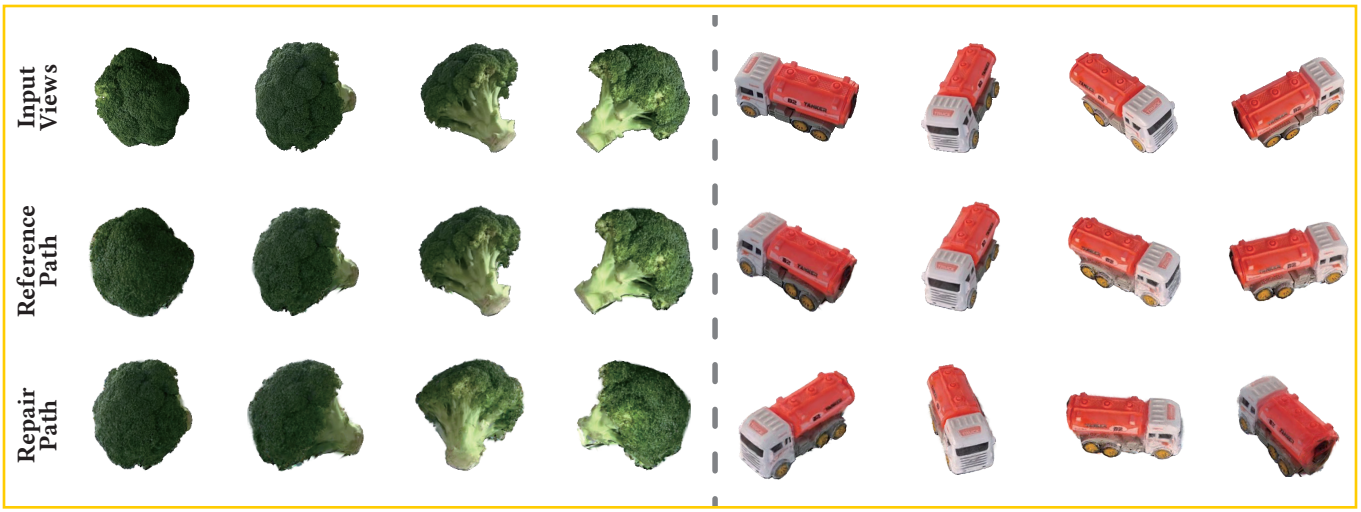

Fig. 18. Qualitative results from reference and repair path. The first row displays the four reference views, the second row shows the renderings from the reference path and the third row shows the renderings from the repair path. GaussianObject achieves high-quality renderings on both reference and repair paths, indicating the effectiveness of our Gaussian repair process.

In the last video set, *videos_4*, we provide input images and results from COLMAP-free experiments using our own collected unposed images. A subfolder named *other_methods* includes videos of reconstruction results obtained by other methods.

Fig. 19. Qualitative examples on the OpenIllumination dataset with four input views.

Fig. 20. Qualitative samples of the Gaussian repaired models on several scenes from different views.

Table 10. Comparisons of per-scene metrics of MipNeRF360 with 4 input views.

Method	bonsai			garden			kitchen		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO [Sun et al. 2022]	0.3324	10.45	0.6980	0.0559	21.75	0.9652	0.3447	10.97	0.7104
3DGS [Kerbl et al. 2023]	0.1408	16.42	0.8458	0.0417	26.05	0.9769	0.1414	18.47	0.8746
DietNeRF [Jain et al. 2021]	0.1333	16.30	0.8682	0.0231	28.63	0.9780	0.1787	11.77	0.8453
RegNeRF [Niemeyer et al. 2022]	0.2736	9.99	0.7847	0.0300	20.89	0.9794	0.3094	9.89	0.7788
FreeNeRF [Yang et al. 2023]	0.2172	10.16	0.8052	0.0300	20.89	0.9760	0.2578	10.08	0.7789
SparseNeRF [Guangcong et al. 2023]	0.2148	10.08	0.8037	0.0618	18.36	0.9556	0.2562	10.04	0.7769
ZeroRF [Shi et al. 2024b]	0.2206	10.36	0.7810	0.0434	22.52	0.9596	0.3324	9.62	0.7157
FSGS [Zhu et al. 2024]	0.1258	17.96	0.8707	0.0279	25.84	0.9766	0.1317	19.40	0.8818
GaussianObject	0.0690	21.51	0.9113	0.0121	30.56	0.9833	0.0687	22.36	0.9104

Table 11. Comparisons of per-scene metrics of MipNeRF360 with 6 input views.

Method	bonsai			garden			kitchen		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO [Sun et al. 2022]	0.3908	10.11	0.6531	0.0416	21.90	0.9728	0.3677	10.88	0.6769
3DGS [Kerbl et al. 2023]	0.1133	18.42	0.8736	0.0330	27.54	0.9809	0.1051	20.40	0.8858
DietNeRF [Jain et al. 2021]	0.0863	22.39	0.9085	0.0274	20.89	0.9760	0.0951	22.80	0.9012
RegNeRF [Niemeyer et al. 2022]	0.2736	9.34	0.7736	0.0300	20.88	0.9794	0.3180	10.00	0.7726
FreeNeRF [Yang et al. 2023]	0.0904	22.09	0.9083	0.0300	20.89	0.9760	0.0847	23.78	0.9153
SparseNeRF [Guangcong et al. 2023]	0.2530	9.45	0.7695	0.0395	20.82	0.9738	0.2997	10.01	0.7516
ZeroRF [Shi et al. 2024b]	0.1038	20.86	0.8992	0.0190	31.77	0.9829	0.1264	19.80	0.8813
FSGS [Zhu et al. 2024]	0.1100	19.69	0.8902	0.0274	25.48	0.9778	0.0932	22.86	0.9112
GaussianObject	0.0499	23.53	0.9313	0.0104	31.97	0.9864	0.0487	25.50	0.9358

Table 12. Comparisons of per-scene metrics of MipNeRF360 with 9 input views.

Method	bonsai			garden			kitchen		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO [Sun et al. 2022]	0.3658	11.17	0.6938	0.0367	22.09	0.9747	0.3672	10.95	0.6843
3DGS [Kerbl et al. 2023]	0.0932	20.68	0.9004	0.0222	29.46	0.9839	0.0771	22.74	0.9149
DietNeRF [Jain et al. 2021]	0.0706	24.45	0.9263	0.0274	20.89	0.9760	0.0774	25.30	0.9247
RegNeRF [Niemeyer et al. 2022]	0.2875	9.45	0.7706	0.0300	20.89	0.9794	0.2735	10.70	0.8050
FreeNeRF [Yang et al. 2023]	0.0788	23.99	0.9263	0.0164	33.05	0.9859	0.0702	25.96	0.9335
SparseNeRF [Guangcong et al. 2023]	0.2857	9.38	0.7447	0.0298	23.04	0.9776	0.3313	10.67	0.7481
ZeroRF [Shi et al. 2024b]	0.0657	24.72	0.9312	0.0174	32.75	0.9841	0.0770	25.88	0.9227
FSGS [Zhu et al. 2024]	0.0816	22.72	0.9164	0.0198	29.28	0.9815	0.0804	23.91	0.9213
GaussianObject	0.0382	25.65	0.9502	0.0089	33.09	0.9886	0.0353	27.11	0.9526

Table 13. Comparisons of per-scene metrics of OmniObject3D with 4 input views.

Method	backpack_016			box_043			broccoli_003		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.2674	11.41	0.7794	0.0814	20.21	0.9455	0.0953	16.14	0.9229
3DGS	0.1542	15.13	0.8505	0.0536	18.57	0.9665	0.0677	15.68	0.9458
DietNeRF	0.2903	11.30	0.8183	0.1866	15.54	0.8867	0.1028	14.65	0.9172
RegNeRF	0.2987	9.56	0.8022	0.1298	16.40	0.9455	0.1109	14.08	0.9239
FreeNeRF	0.1358	11.57	0.8899	0.0775	17.59	0.9516	0.0570	16.36	0.9559
SparseNeRF	0.2570	9.78	0.7985	0.0960	16.52	0.9437	0.1068	14.18	0.9208
ZeroRF	0.0528	25.13	0.9459	0.0211	33.94	0.9827	0.0228	30.32	0.9745
FSGS	0.1257	19.48	0.9089	0.0835	23.70	0.9590	0.0399	23.25	0.9670
GaussianObject	0.0425	25.61	0.9511	0.0140	33.22	0.9833	0.0138	30.59	0.9781

Method	corn_007			dinosaur_006			flower_pot_007		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1135	21.81	0.9279	0.1924	16.08	0.8539	0.1281	15.04	0.8951
3DGS	0.0826	18.91	0.9496	0.1124	18.31	0.8952	0.0741	14.79	0.9330
DietNeRF	0.0244	33.80	0.9729	0.1168	16.07	0.9161	0.0482	16.22	0.9539
RegNeRF	0.1869	16.65	0.9200	0.2371	14.60	0.8838	0.1499	13.35	0.9065
FreeNeRF	0.0732	24.35	0.9517	0.1071	16.01	0.9197	0.0962	15.45	0.9259
SparseNeRF	0.1678	16.77	0.9041	0.1781	15.22	0.8881	0.1379	14.20	0.9139
ZeroRF	0.0343	33.68	0.9694	0.0460	26.59	0.9545	0.0235	30.77	0.9759
FSGS	0.0650	26.37	0.9577	0.0969	21.19	0.9266	0.0363	26.88	0.9659
GaussianObject	0.0166	34.44	0.9781	0.0349	27.68	0.9607	0.0156	31.22	0.9795

Method	gloves_009			guitar_002			hamburger_012		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.2285	12.45	0.8229	0.1178	15.46	0.9102	0.1510	19.00	0.9028
3DGS	0.1303	13.04	0.8945	0.1130	12.74	0.8995	0.0954	16.24	0.9274
DietNeRF	0.1222	14.08	0.9140	0.0917	24.20	0.9342	0.2062	15.46	0.8675
RegNeRF	0.1590	12.90	0.8989	0.1874	12.52	0.8660	0.1654	15.60	0.9154
FreeNeRF	0.1458	13.50	0.8951	0.1004	13.44	0.9206	0.0790	17.33	0.9442
SparseNeRF	0.1440	13.03	0.8912	0.3535	10.53	0.7153	0.1716	15.68	0.8914
ZeroRF	0.1578	14.61	0.8793	0.0785	20.87	0.9373	0.0355	29.33	0.9634
FSGS	0.1427	16.80	0.9140	0.0416	27.92	0.9659	0.0747	22.20	0.9474
GaussianObject	0.0297	26.94	0.9695	0.0141	32.37	0.9816	0.0224	31.46	0.9694

Table 14. Comparisons of per-scene metrics of OmniObject3D with 4 input views. *continued*

Method	picnic_basket_009			pineapple_013			sandwich_003			suitcase_006		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1540	16.08	0.8887	0.1677	14.39	0.8747	0.0423	25.15	0.9787	0.1865	14.52	0.8697
3DGS	0.0837	19.60	0.9167	0.0573	16.74	0.9484	0.0285	22.44	0.9850	0.1028	15.18	0.9331
DietNeRF	0.1512	15.02	0.8940	0.0933	16.48	0.9336	0.0253	22.52	0.9851	0.0542	26.60	0.9639
RegNeRF	0.1313	14.73	0.9303	0.1431	14.01	0.9005	0.0293	22.52	0.9875	0.2636	12.03	0.8587
FreeNeRF	0.0772	15.82	0.9433	0.0638	16.74	0.9471	0.0293	22.52	0.9850	0.0755	25.66	0.9422
SparseNeRF	0.3376	14.99	0.8717	0.1918	16.01	0.9113	0.0478	20.47	0.9750	0.2487	12.55	0.8384
ZeroRF	0.0391	28.33	0.9648	0.0569	22.81	0.9413	0.0171	30.44	0.9886	0.0520	23.83	0.9611
FSGS	0.0417	25.60	0.9612	0.0608	20.87	0.9447	0.0110	33.34	0.9900	0.0682	23.70	0.9528
GaussianObject	0.0220	31.44	0.9733	0.0208	28.16	0.9676	0.0072	34.34	0.9919	0.0259	28.69	0.9758

Method	timer_010			toy_plane_005			toy_truck_037			vase_012		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1104	18.90	0.9345	0.0929	22.37	0.9423	0.1415	16.34	0.9075	0.1912	16.01	0.8612
3DGS	0.0940	17.41	0.9407	0.0470	22.94	0.9672	0.0567	18.26	0.9583	0.1086	17.96	0.8968
DietNeRF	0.1411	17.30	0.9143	0.0321	24.12	0.9752	0.1059	17.73	0.9382	0.1856	14.41	0.8637
RegNeRF	0.1557	16.79	0.9310	0.0354	24.12	0.9810	0.1531	15.37	0.9240	0.3109	13.18	0.8799
FreeNeRF	0.0616	18.80	0.9664	0.0354	24.12	0.9752	0.0612	18.28	0.9562	0.1309	14.74	0.9130
SparseNeRF	0.1742	17.37	0.9317	0.0655	22.32	0.9603	0.1257	15.55	0.9161	0.1664	13.63	0.8939
ZeroRF	0.0258	32.97	0.9848	0.0228	30.52	0.9827	0.0251	30.68	0.9758	0.0440	27.51	0.9634
FSGS	0.0362	29.15	0.9782	0.0256	29.93	0.9798	0.0517	26.15	0.9588	0.0607	23.53	0.9490
GaussianObject	0.0152	33.43	0.9867	0.0102	34.30	0.9884	0.0137	32.23	0.9820	0.0327	29.10	0.9676

Table 15. Comparisons of per-scene metrics of OmniObject3D with 6 input views.

Method	backpack_016			box_043			broccoli_003		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.2363	12.56	0.8040	0.1028	19.74	0.9412	0.0853	18.16	0.9385
3DGS	0.0901	21.26	0.9122	0.0582	18.16	0.9607	0.0483	16.77	0.9621
DietNeRF	0.2117	11.25	0.8546	0.1111	17.25	0.9325	0.0994	15.03	0.9267
RegNeRF	0.2818	10.08	0.8291	0.1251	15.85	0.9420	0.0483	16.84	0.9663
FreeNeRF	0.2255	12.30	0.8181	0.0530	18.68	0.9656	0.0483	16.84	0.9625
SparseNeRF	0.4828	11.35	0.7691	0.1030	18.16	0.9602	0.0483	16.84	0.9625
ZeroRF	0.0494	27.87	0.9529	0.0195	36.07	0.9843	0.0212	32.54	0.9768
FSGS	0.0815	22.34	0.9308	0.0266	32.41	0.9799	0.1163	20.96	0.9473
GaussianObject	0.0318	27.60	0.9593	0.0121	35.15	0.9868	0.0108	32.78	0.9842

Method	corn_007			dinosaur_006			flower_pot_007		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1056	22.43	0.9540	0.1456	17.91	0.8879	0.1348	15.36	0.8890
3DGS	0.0796	19.22	0.9551	0.1030	19.04	0.8968	0.0675	15.43	0.9392
DietNeRF	0.0225	35.23	0.9775	0.1905	14.69	0.8592	0.0482	16.22	0.9539
RegNeRF	0.1444	17.56	0.9413	0.2226	14.38	0.8905	0.1323	13.50	0.9122
FreeNeRF	0.0794	19.44	0.9523	0.0970	16.26	0.9275	0.0525	16.22	0.9539
SparseNeRF	0.0793	19.42	0.9523	0.3566	15.12	0.8636	0.1861	12.72	0.8600
ZeroRF	0.0309	34.98	0.9726	0.0409	29.03	0.9633	0.0217	32.83	0.9791
FSGS	0.0511	29.35	0.9634	0.0503	26.38	0.9577	0.0386	27.17	0.9658
GaussianObject	0.0135	35.77	0.9821	0.0256	29.65	0.9708	0.0107	34.40	0.9866

Method	gloves_009			guitar_002			hamburger_012		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1373	15.00	0.9011	0.0803	17.61	0.9368	0.1697	18.75	0.9068
3DGS	0.1284	13.42	0.8967	0.1176	12.50	0.8944	0.0949	16.48	0.9278
DietNeRF	0.1647	13.92	0.8797	0.0401	28.63	0.9650	0.1563	16.14	0.9029
RegNeRF	0.1919	13.12	0.8849	0.1228	14.95	0.9105	0.2122	14.52	0.9015
FreeNeRF	0.0905	14.39	0.9331	0.0272	33.19	0.9767	0.0790	17.33	0.9442
SparseNeRF	0.1672	13.14	0.8744	0.3992	10.46	0.7581	0.5058	13.82	0.8368
ZeroRF	0.0352	30.52	0.9728	0.0669	23.41	0.9423	0.0292	33.60	0.9710
FSGS	0.0930	20.34	0.9333	0.0509	26.17	0.9627	0.0512	27.72	0.9568
GaussianObject	0.0214	30.75	0.9788	0.0116	34.24	0.9861	0.0152	34.29	0.9801

Table 16. Comparisons of per-scene metrics of OmniObject3D with 6 input views. *continued*

Method	picnic_basket_009			pineapple_013			sandwich_003			suitcase_006		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1371	17.15	0.9021	0.1448	17.23	0.8945	0.0793	25.23	0.9663	0.1489	15.70	0.9050
3DGS	0.0490	26.20	0.9494	0.0682	16.16	0.9388	0.0374	21.32	0.9784	0.1026	15.19	0.9331
DietNeRF	0.1264	15.24	0.9123	0.0564	16.74	0.9472	0.0253	22.52	0.9851	0.0373	29.00	0.9750
RegNeRF	0.0772	15.82	0.9524	0.1467	14.15	0.8969	0.0293	22.52	0.9875	0.1351	15.80	0.9186
FreeNeRF	0.0772	15.82	0.9433	0.0638	16.74	0.9471	0.0293	22.52	0.9850	0.0682	26.99	0.9469
SparseNeRF	0.3520	15.91	0.9024	0.1997	15.63	0.9061	0.0591	20.90	0.9734	0.2311	15.50	0.8714
ZeroRF	0.0337	31.11	0.9698	0.0257	29.64	0.9671	0.0058	38.85	0.9958	0.0513	26.33	0.9676
FSGS	0.0283	31.43	0.9742	0.0299	28.42	0.9697	0.0079	37.49	0.9938	0.1395	18.73	0.9197
GaussianObject	0.0185	33.30	0.9779	0.0146	31.03	0.9797	0.0041	38.29	0.9954	0.0192	30.54	0.9826

Method	timer_010			toy_plane_005			toy_truck_037			vase_012		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1335	19.63	0.9395	0.0919	21.98	0.9451	0.1244	18.42	0.9176	0.1343	18.49	0.9125
3DGS	0.0751	18.15	0.9556	0.0559	22.24	0.9639	0.0583	18.23	0.9581	0.0822	21.16	0.9204
DietNeRF	0.1221	16.94	0.9225	0.0321	24.12	0.9752	0.0988	17.53	0.9335	0.2242	13.73	0.8517
RegNeRF	0.1444	16.39	0.9410	0.0354	24.12	0.9810	0.1553	15.88	0.9187	0.2404	13.07	0.8783
FreeNeRF	0.0616	18.80	0.9664	0.0354	24.12	0.9752	0.0612	18.28	0.9562	0.0952	15.51	0.9354
SparseNeRF	0.1266	17.09	0.9349	0.0569	23.93	0.9688	0.1290	15.97	0.9122	0.2075	13.66	0.8827
ZeroRF	0.0246	34.57	0.9860	0.0128	36.87	0.9896	0.0218	33.79	0.9801	0.0377	31.05	0.9716
FSGS	0.0582	25.42	0.9655	0.0207	31.97	0.9838	0.1065	19.30	0.9370	0.0782	22.55	0.9478
GaussianObject	0.0139	34.73	0.9885	0.0075	36.59	0.9918	0.0080	35.40	0.9885	0.0253	31.69	0.9763

Table 17. Comparisons of per-scene metrics of OmniObject3D with 9 input views.

Method	backpack_016			box_043			broccoli_003		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.2299	12.43	0.8179	0.0664	22.20	0.9677	0.0683	21.03	0.9618
3DGS	0.0519	26.80	0.9494	0.0506	18.65	0.9669	0.0531	16.45	0.9552
DietNeRF	0.1806	11.48	0.8684	0.0481	18.68	0.9656	0.0995	14.19	0.9211
RegNeRF	0.2967	10.46	0.8276	0.1051	17.39	0.9490	0.0483	16.84	0.9663
FreeNeRF	0.2422	12.39	0.8084	0.0530	18.68	0.9656	0.0483	16.84	0.9625
SparseNeRF	0.4932	10.46	0.7481	0.0615	18.59	0.9639	0.2476	13.34	0.8900
ZeroRF	0.0459	30.15	0.9582	0.0184	37.80	0.9857	0.0216	33.42	0.9775
FSGS	0.1250	14.84	0.8963	0.0226	33.72	0.9858	0.0169	33.03	0.9817
GaussianObject	0.0229	31.41	0.9737	0.0089	37.98	0.9910	0.0085	34.91	0.9882

Method	corn_007			dinosaur_006			flower_pot_007		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.0921	22.05	0.9598	0.0985	19.56	0.9269	0.0987	19.15	0.9319
3DGS	0.0801	19.11	0.9510	0.0620	23.57	0.9377	0.0622	15.65	0.9438
DietNeRF	0.0184	36.58	0.9824	0.2514	14.57	0.8359	0.0482	16.22	0.9539
RegNeRF	0.0870	20.45	0.9546	0.1914	14.51	0.8951	0.0525	16.22	0.9598
FreeNeRF	0.0221	36.42	0.9779	0.0970	16.26	0.9275	0.0525	16.22	0.9539
SparseNeRF	0.2863	18.94	0.9340	0.4706	13.86	0.8191	0.1816	13.70	0.8855
ZeroRF	0.0352	35.69	0.9715	0.0393	30.28	0.9679	0.0199	34.60	0.9816
FSGS	0.0280	34.01	0.9792	0.0533	27.19	0.9593	0.0295	30.55	0.9750
GaussianObject	0.0089	38.33	0.9879	0.0201	31.61	0.9780	0.0087	36.13	0.9896

Method	gloves_009			guitar_002			hamburger_012		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.0879	17.36	0.9425	0.0481	21.24	0.9614	0.0896	23.11	0.9592
3DGS	0.1239	13.78	0.9039	0.1060	12.90	0.9060	0.0937	16.56	0.9282
DietNeRF	0.1353	13.77	0.8923	0.0318	32.15	0.9730	0.1504	16.01	0.9007
RegNeRF	0.0905	14.39	0.9428	0.0928	16.97	0.9268	0.0790	17.33	0.9567
FreeNeRF	0.1393	13.64	0.8944	0.0236	35.56	0.9818	0.0790	17.33	0.9442
SparseNeRF	0.1584	13.29	0.8768	0.2553	19.14	0.8907	0.4415	15.47	0.8795
ZeroRF	0.0349	31.94	0.9753	0.0768	20.01	0.9370	0.0314	34.14	0.9713
FSGS	0.0817	21.46	0.9422	0.0436	28.66	0.9693	0.0259	33.34	0.9781
GaussianObject	0.0180	32.89	0.9838	0.0100	35.55	0.9888	0.0103	37.09	0.9875

Table 18. Comparisons of per-scene metrics of OmniObject3D with 9 input views. *continued*

Method	picnic_basket_009			pineapple_013			sandwich_003			suitcase_006		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1501	16.50	0.9012	0.0927	18.63	0.9393	0.0509	25.44	0.9818	0.0965	17.72	0.9430
3DGS	0.0232	32.98	0.9768	0.0573	16.73	0.9483	0.0316	21.91	0.9825	0.1026	15.19	0.9331
DietNeRF	0.1416	13.74	0.8889	0.1247	14.14	0.8914	0.0253	22.52	0.9851	0.0325	30.99	0.9799
RegNeRF	0.1746	13.91	0.8965	0.0638	16.74	0.9534	0.0293	22.52	0.9875	0.0317	32.63	0.9903
FreeNeRF	0.0772	15.82	0.9433	0.0638	16.74	0.9471	0.0293	22.52	0.9850	0.0510	30.81	0.9699
SparseNeRF	0.4269	14.14	0.8208	0.4313	14.55	0.8322	0.0448	22.50	0.9843	0.0438	31.75	0.9803
ZeroRF	0.0320	33.12	0.9735	0.0254	30.79	0.9692	0.0057	39.80	0.9961	0.0463	28.34	0.9722
FSGS	0.0393	28.61	0.9712	0.0186	31.83	0.9807	0.0076	35.88	0.9945	0.0452	29.53	0.9746
GaussianObject	0.0139	35.27	0.9836	0.0109	33.00	0.9858	0.0029	41.15	0.9971	0.0143	32.99	0.9876
Method	timer_010			toy_plane_005			toy_truck_037			vase_012		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1811	17.27	0.9222	0.2574	16.89	0.8464	0.1297	17.93	0.9270	0.1154	18.89	0.9242
3DGS	0.0450	28.44	0.9669	0.0499	22.88	0.9650	0.0557	18.27	0.9584	0.0566	24.65	0.9486
DietNeRF	0.1010	17.56	0.9359	0.0321	24.12	0.9752	0.1074	17.25	0.9337	0.2255	13.52	0.8559
RegNeRF	0.0616	18.80	0.9725	0.0354	24.12	0.9810	0.0612	18.28	0.9647	0.2283	13.26	0.8901
FreeNeRF	0.0616	18.80	0.9664	0.0354	24.12	0.9752	0.0612	18.28	0.9562	0.0952	15.51	0.9354
SparseNeRF	0.1180	17.47	0.9353	0.0354	24.10	0.9752	0.1425	15.93	0.9067	0.2013	14.46	0.8883
ZeroRF	0.0231	35.46	0.9877	0.0124	37.76	0.9902	0.0215	34.09	0.9805	0.0366	32.43	0.9742
FSGS	0.0427	28.50	0.9804	0.0204	32.56	0.9829	0.0686	23.09	0.9565	0.0406	28.96	0.9732
GaussianObject	0.0106	36.87	0.9921	0.0058	38.48	0.9941	0.0072	36.72	0.9905	0.0219	32.94	0.9799

Table 19. Comparisons of per-scene metrics of OpenIllumination with 4 input views. Methods with †means the metrics are from the ZeroRF paper [Shi et al. 2024b].

Method	stone			pumpkin			toy			potato		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.0829	21.99	0.8957	0.0945	23.07	0.9370	0.0928	21.69	0.9121	0.1196	21.37	0.9097
3DGS	0.2890	10.93	0.8107	0.2950	11.43	0.8692	0.2898	11.16	0.8213	0.3063	12.03	0.8639
DietNeRF†	0.0850	24.05	0.9210	0.0600	26.54	0.9700	0.0790	24.98	0.9490	0.1030	23.00	0.9490
RegNeRF†	0.4830	10.26	0.6020	0.4650	11.74	0.7490	0.4760	10.04	0.6370	0.5050	11.63	0.7190
FreeNeRF†	0.2100	12.91	0.7790	0.3120	11.54	0.8270	0.3510	10.79	0.7860	0.4610	11.70	0.7960
SparseNeRF	0.2600	12.99	0.8315	0.3029	11.44	0.7991	0.1181	13.67	0.9200	0.2821	13.26	0.8798
ZeroRF†	0.0720	25.07	0.9180	0.0750	26.07	0.9610	0.0890	23.72	0.9360	0.0960	26.27	0.9460
GaussianObject	0.0542	23.96	0.9204	0.0538	25.89	0.9579	0.0493	24.95	0.9453	0.0690	25.09	0.9424
Method	pine			shroom			cow			cake		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.1177	19.04	0.8791	0.1754	18.13	0.8533	0.1854	18.14	0.8523	0.0790	25.74	0.9396
3DGS	0.3038	10.07	0.8095	0.3594	11.03	0.8182	0.3157	11.81	0.8573	0.2475	13.54	0.9133
DietNeRF†	0.0930	20.94	0.9240	0.1660	19.91	0.9110	0.2070	16.30	0.8940	0.0600	28.97	0.9710
RegNeRF†	0.4860	9.37	0.5710	0.5510	10.66	0.6580	0.4600	11.99	0.7480	0.3590	17.21	0.8680
FreeNeRF†	0.2200	10.17	0.7910	0.5540	11.46	0.7510	0.4580	11.18	0.7460	0.2990	17.95	0.8990
SparseNeRF	0.1412	12.28	0.8981	0.2026	13.33	0.8963	0.2170	13.64	0.9003	0.2584	18.21	0.9214
ZeroRF†	0.1160	20.68	0.9030	0.1340	23.14	0.9120	0.1390	21.91	0.9050	0.0580	29.44	0.9650
GaussianObject	0.0761	21.68	0.9143	0.0872	24.16	0.9211	0.0970	22.81	0.9208	0.0499	28.58	0.9613

Table 20. Comparisons of per-scene metrics of OpenIllumination with 6 input views. Methods with †means the metrics are from the ZeroRF paper [Shi et al. 2024b].

Method	stone			pumpkin			toy			potato		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.0621	24.22	0.9132	0.0682	25.82	0.9527	0.0675	24.44	0.9313	0.0931	24.24	0.9279
3DGS	0.3139	10.87	0.7702	0.2547	12.98	0.8823	0.2993	11.10	0.7943	0.3197	11.70	0.8462
DietNeRF†	0.0850	24.87	0.9210	0.0730	24.80	0.9660	0.0860	25.37	0.9440	0.0870	25.63	0.9550
RegNeRF†	0.2880	13.80	0.8480	0.3500	13.58	0.8480	0.2370	13.54	0.8840	0.3480	13.92	0.8540
FreeNeRF†	0.2360	11.62	0.7910	0.2930	11.71	0.8640	0.3460	10.65	0.8140	0.3970	11.35	0.8320
SparseNeRF	0.2452	12.91	0.8091	0.1468	14.81	0.9406	0.1181	13.67	0.9200	0.3519	12.36	0.8614
ZeroRF†	0.0630	26.30	0.9290	0.0640	27.87	0.9660	0.0620	27.28	0.9500	0.0840	27.26	0.9510
GaussianObject	0.0430	26.07	0.9301	0.0415	28.06	0.9648	0.0402	27.55	0.9535	0.0571	26.56	0.9495

Method	pine			shroom			cow			cake		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
DVGO	0.0924	20.61	0.8976	0.1327	20.88	0.8891	0.1331	22.00	0.8997	0.0571	28.15	0.9562
3DGS	0.2687	11.73	0.8112	0.3633	11.49	0.7671	0.3246	11.55	0.8329	0.2277	14.38	0.9174
DietNeRF†	0.1190	18.16	0.9020	0.1190	23.71	0.9300	0.1330	21.50	0.9300	0.0590	29.58	0.9730
RegNeRF†	0.3370	11.87	0.8070	0.3290	13.22	0.8630	0.4050	13.07	0.8070	0.1280	19.66	0.9580
FreeNeRF†	0.3280	8.85	0.7530	0.5050	10.12	0.7640	0.4420	11.09	0.7840	0.2650	16.33	0.9000
SparseNeRF	0.3807	6.61	0.5537	0.3551	11.92	0.8290	0.2227	13.64	0.8983	0.2840	16.46	0.9100
ZeroRF†	0.0880	22.26	0.9180	0.1060	26.34	0.9280	0.1180	23.74	0.9210	0.0520	31.00	0.9690
GaussianObject	0.0629	23.25	0.9264	0.0712	26.35	0.9307	0.0804	24.43	0.9325	0.0389	30.06	0.9673