LM-Gaussian: Boost Sparse-view 3D Gaussian Splatting with Large Model Priors

Hanyang Yu[1], Xiaoxiao Long[1,‡] and Ping Tan[1]

***Abstract*—We aim to address sparse-view reconstruction of a 3D scene by leveraging priors from large-scale vision models. While recent advancements such as 3D Gaussian Splatting (3DGS) have demonstrated remarkable successes in 3D reconstruction, these methods typically necessitate hundreds of input images that densely capture the underlying scene, making them time-consuming and impractical for real-world applications. However, sparse-view reconstruction is inherently ill-posed and under-constrained, often resulting in inferior and incomplete outcomes. This is due to issues such as failed initialization, overfitting on input images, and a lack of details. To mitigate these challenges, we introduce LM-Gaussian, a method capable of generating high-quality reconstructions from a limited number of images. Specifically, we propose a robust initialization module that leverages stereo priors to aid in the recovery of camera poses and the reliable point clouds. Additionally, a diffusion-based refinement is iteratively applied to incorporate image diffusion priors into the Gaussian optimization process to preserve intricate scene details. Finally, we utilize video diffusion priors to further enhance the rendered images for realistic visual effects. Overall, our approach significantly reduces the data acquisition requirements compared to previous 3DGS methods. We validate the effectiveness of our framework through experiments on various public datasets, demonstrating its potential for high-quality 360-degree scene reconstruction. Visual results are on our website(lm-gaussian.github.io).**

***Index Terms*—sparse-view, reconstruction, gaussian splatting, large models.**

I. Introduction

3D scene reconstruction and novel view synthesis from sparse-view images present significant challenges in the field of computer vision. Recent advancements in neural radiance fields (NeRF) [51] and 3D Gaussian splatting (3DGS) [30] have made notable progresses in synthesizing novel views, but they typically require hundreds of images to reconstruct a scene. Capturing such a dense set of images is often impractical, raising the inconvenience for utilizing these technologies. Although efforts have been made to address sparse-view settings, existing works are still limited to straightforward facing scenarios, such as the LLFF dataset [50], which involve small-angle rotations and simple orientations. For large-scale 360-degree scenes, the problems of being ill-posed and under-constrained hinder the employment of these methods. In this work, we present a new method that is capable of producing high-quality reconstruction from sparse input images, demonstrating promising results even in challenging 360-degree scenes.

‡ Xiaoxiao Long is the corresponding author (xxlong@connect.hku.hk).
[1] The Hong Kong University of Science and Technology
Submitted for review on Sep. 4th, 2024.

There are three main obstacles that prevent 3D Gaussian splatting from achieving high-quality 3D reconstruction with sparse-view images. 1) **Failed initialization:** 3DGS heavily relies on pre-calculated camera poses and point clouds for initializing Gaussian spheres. However, traditional Structure-from-Motion (SfM) techniques [60] cannot successfully handle the sparse-view setting due to insufficient overlap among the input images, therefore yielding inaccurate camera poses and unreliable point clouds for 3DGS initialization. 2) **Overfitting on input images:** Lacking sufficient images to provide constraints, 3DGS tends to be overfitted on the sparse input images and therefore produces novel synthesized views with severe artifacts. 3) **Lack of details:** Given limited multi-view constraints and geometric cues, 3DGS always fails to recover the details of the captured 3D scene and the unobserved regions, which significantly degrades the final reconstruction quality.

To tackle these challenges, we introduce LM-Gaussian, a novel method capable of producing high-quality reconstructions from sparse input images by incorporating large model priors. The key idea is leveraging the power of various large model priors to boost the reconstruction of 3D gaussian splatting with three primary objectives: 1) **Robust initialization**; 2) **Overfitting prevention**; 3) **Detail preservation**.

For robust initialization, instead of relying on traditional SfM methods [59], [60], we propose a novel initialization module utilizing stereo priors from DUSt3R [73]. DUSt3R is a comprehensive stereo model that takes pairs of images as input and directly generates corresponding 3D point clouds. Through a global optimization process, it derives camera poses from the input images and establishes a globally registered point cloud. However, the global point cloud often exhibits artifacts and floaters in background regions due to the inherent bias of DUSt3R towards foreground regions. To mitigate this issue, we introduce a Background-Aware Depth-guided Initialization module. Initially, we use depth priors to refine the point clouds produced by DUSt3R, particularly in the background areas of the scene. Additionally, we employ iterative filtering operations to eliminate unreliable 3D points by conducting geometric consistency checks and confidence-based evaluations. This approach ensures the generation of a clean and reliable 3D point cloud for initializing 3D Gaussian splatting.

Once a robust initialization is obtained, photo-metric loss is commonly used to optimize 3D Gaussian spheres. However, in the sparse-view setting, solely using photo-metric loss will make 3DGS overfit on input images. To address this issue, we introduce multiple geometric constraints to regularize the optimization of 3DGS effectively. Firstly, a multi-scale depth

regularization term is incorporated to encourage 3DGS to capture both local and global geometric structures of depth priors. Secondly, a cosine-constrained normal regularization term is introduced to ensure that the geometric variations of 3DGS to be aligned with normal priors. Lastly, a weighted virtual-view regularization term is applied to enhance the resilience of 3DGS to unseen view directions.

To preserve intricate scene details, we introduce Iterative Gaussian Refinement Module, which leverages diffusion priors to recover high-frequency details. We leverage a diffusion-based Gaussian repair model to restore the images rendered from 3DGS, aiming to enhance image details with good visual effects. The enhanced images are used as additional pseudo ground-truth to optimize 3DGS. Such a refinement operation is iteratively employed in 3DGS optimization, which gradually inject the image diffusion priors into 3DGS for detail enhancement. Specifically, the Gaussian repair model is built on ControlNet with injected Lora layers, where the sparse input images are used to finetune the Lora layers so that repair model could work well on specific scenes.

By combining the strengths of different large model priors, LM-Gaussian can synthesize new views with competitive quality and superior details compared to state-of-the-art methods in sparse-view settings, particularly in 360-degree scenes. The contributions of our method can be summarized as follows:

- We propose a new method capable of generating high-quality novel views in a sparse-view setting with large model priors. Our method surpasses recent works in sparse-view settings, especially in large-scale 360-degree scenes.
- We introduce a Background-Aware Depth-guided Initialization Module, capable of simultaneously reconstructing high-quality dense point clouds and camera poses for initialization.
- We introduce a Multi-modal Regularized Gaussian Reconstruction Module that leverages regularization techniques from various domains to avoid overfitting issues.
- We present an Iterative Gaussian Refinement Module which uses diffusion priors to recover scene details and achieve high-quality novel view synthesis results.

II. Related work

3D Representations for Novel-view synthesis: Novel view synthesis (NVS) involves rendering unseen viewpoints of a scene from a given set of images. One popular approach is Neural Radiance Fields (NeRF), which uses a Multilayer Perceptron (MLP) to represent 3D scenes and renders via volume rendering. Several works have aimed to enhance NeRF's performance by addressing aspects such as speed [17], [18], [35], [54], quality [3], [68], [72], [95], and adapting it to novel tasks [58], [65], [76], [98]. While NeRF relies on a neural network to represent the radiance field, 3D Gaussian Splatting (3DGS) [30] stands out by using an ensemble of anisotropic 3D Gaussians to represent the scene and employs differentiable splatting for rendering. This approach has shown remarkable success in efficiently and accurately reconstructing complex real-world scenes with superior quality. Recent works have further extended the capabilities of 3DGS to perform various downstream tasks, including text-to-3D generation [11], [38], [67], [82], [85], [89], [101], dynamic scene representation [2], [25], [36], [42], [43], [46], [47], [61], [66], [78], [91], [93], [102], editing [9], [71], [87], [100], compression [16], [33], [53], [70], SLAM [24], [29], [48], [83], animating humans [1], [21], [22], [37], [44], [52], [56], [62], [99], [104], and other novel tasks [12], [19], [23], [27], [39], [41], [45], [57], [63], [80], [84], [100], [102], [105].

Sparse View Scene Reconstruction and Synthesis: Sparse view reconstruction aims to reconstruct a scene using a limited number of input views. Existing methods can be classified into regularization techniques and generalizable reconstruction priors. Several works [15], [34], [55], [69], [86] address this challenge by employing strategies like depth regularization and frequency regularization. Some approaches [26], [79], [81] utilize pre-trained models as a regularization mechanism to guide training based on established knowledge. Another research direction [8], [90], [103] focuses on training priors to synthesize novel views across diverse scenes. Building on the achievements of 3D Gaussian Splatting, recent methods such as SparseGS [69], pixelSplat [7], and MVSplat [10] leverage stereo view interpolation to facilitate training. GeNVS [6] and latentSplat [77] utilize rendering view-conditioned feature fields followed by 2D generative decoding to generate novel views. While these methods have demonstrated improved results in new view synthesis, they still encounter challenges in producing clear views under high uncertainty and may encounter difficulties in handling 360-degree scenes.

Unposed Scene Reconstruction: The methods mentioned above all rely on known camera poses, and Structure from Motion (SfM) algorithms often struggle to predict camera poses and point clouds with sparse inputs, mainly due to a lack of image correspondences. Therefore, removing camera parameter preprocessing is another active line of research. For instance, iNeRF [88] demonstrates that poses for new view images can be estimated using a reconstructed NeRF model. NeRFmm [75] concurrently optimizes camera intrinsics, extrinsics, and NeRF training. BARF [40] introduces a coarse-to-fine positional encoding strategy for joint optimization of camera poses and NeRF. GARF [13] illustrates that utilizing Gaussian-MLPs simplifies and enhances the accuracy of joint pose and scene optimization. Recent works like Nope-NeRF [5], LocalRF [49], and CF-3DGS [92] leverage depth information to constrain NeRF or 3DGS optimization. While demonstrating promising outcomes on forward-facing datasets such as LLFF [50], these methods encounter challenges when dealing with complex camera trajectories involving significant camera motion, such as 360-degree large-scale scenes.

III. Preliminary

A. 3D Gaussian Splatting

3D Gaussian Splatting (3D-GS) represents a 3D scene with a set of 3D Gaussians. Specifically, a Gaussian primitive can be defined by a center $\mu \in \mathbb{R}^3$, a scaling factor $s \in \mathbb{R}^3$, and a

rotation quaternion $q \in \mathbb{R}^4$. Each 3D Gaussian is characterized by:

$$G(x) = \frac{1}{(2\pi)^{3/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \quad (1)$$

where the covariance matrix Σ can be derived from the scale s and rotation q.

To render an image from a specified viewpoint, the color of each pixel p is computed by blending K ordered Gaussians $\{G_i \mid i = 1, \cdots, K\}$ that overlap with p using the following blending equation:

$$c(p) = \sum_{i=1}^{K} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (2)$$

where α_i is determined by evaluating a projected 2D Gaussian from G_i at p multiplied by a learned opacity of G_i, and c_i represents the learnable color of G_i. The Gaussians covering p are sorted based on their depths under the current viewpoint. Leveraging differentiable rendering techniques, all attributes of the Gaussians can be optimized end-to-end through training for view reconstruction.

Rasterizing Depth for Gaussians: Following the depth calculation approach introduced in RaDe-GS [94], the center μ_i of a Gaussian G_i is initially projected into the camera coordinate system as μ_i'. Upon obtaining the center value (x_i', y_i', z_i') for each Gaussian, the depth (x, y, z) of each pixel is computed as:

$$d = z_i' + \mathbf{p} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}, \mu_i' = \begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = \mathbf{W}\mu_i + \mathbf{t}, \quad (3)$$

where z_i' represents the depth of the Gaussian center, $\Delta x = x_i' - x$ and $\Delta y = y_i' - y$ denote the relative pixel positions. The vector $\mathbf{p}$ is determined by the Gaussian parameters $[\mathbf{W}, \mathbf{t}] \in \mathbb{R}^{3\times4}$.

Rasterizing Normal for Gaussians: In accordance with RaDe-GS, the normal direction of the projected Gaussian is aligned with the plane's normal. To compute the normal map, we transform the normal vector from the 'rayspace' to the 'camera space' as follows:

$$\mathbf{n} = -\mathbf{J}^\top \begin{pmatrix} \frac{\mathbf{z_i'}}{\mathbf{z_i}}\mathbf{p} & 1 \end{pmatrix}^\top, \quad (4)$$

where $\mathbf{J}$ represents the local affine matrix, and the vector $\mathbf{p}$ has been defined earlier.

B. Diffusion model

In recent years, diffusion models have emerged as the state-of-the-art approach for image synthesis. These models are characterized by a predefined forward noising process $\{\mathbf{z_t}\}_{t=1}^T$ that progressively corrupts the data by introducing random noise ϵ.

$$z_t = \sqrt{\overline{\alpha}_t} z_0 + \sqrt{1 - \overline{\alpha}_t}\epsilon, \epsilon \in \mathbf{N}(\mathbf{0}, \mathbf{I}), \quad (5)$$

where $t \in [1, T]$ denotes the time step and $\overline{\alpha}_t = \alpha_1 \cdot \alpha_2 ... \alpha_t$ represents a decreasing sequence. These models can generate samples from the underlying data distribution given pure noise by training a neural network to learn a reversed denoising process. Having learned from hundreds of millions of images from the internet, diffusion priors exhibit a remarkable capacity to recover real-world details.

TABLE I
SYMBOL DEFINITION. FOR CLARITY, WE FIRST DEFINITION SYMBOLS MENTIONED IN THIS PAPER.

Symbol	Definition
$\boldsymbol{I}_k$	RGB input image of k_{th} view
$\boldsymbol{P}_k$	3D point map of k_{th} view
$\boldsymbol{\eta}_k$	Confidence map of k_{th} view
$\hat{\boldsymbol{D}}_{\boldsymbol{k}}$, $\hat{\boldsymbol{N}}_{\boldsymbol{k}}$	Monocular estimated depth / normal map of k_{th} view
$\bar{\boldsymbol{I}}_{\boldsymbol{k}}$, $\bar{\boldsymbol{D}}_{\boldsymbol{k}}$, $\bar{\boldsymbol{N}}_{\boldsymbol{k}}$	Gaussian-rendered RGB / depth / normal image in k_{th} view.

IV. METHOD

A. Overview

In this paper, we introduce a new method called LM-Gaussian, which aims to generate high-quality novel views of 360-degree scenes using a limited number of input images. Our approach integrates multiple large model priors and is composed of four key modules: 1) **Background-Aware Depth-guided Initialization**: This module extends DUSt3R for camera pose estimation and detailed 3D point cloud creation. By integrating depth priors and point cleaning, we achieve a high-quality point cloud for Gaussian initialization (see Section IV-B). 2) **Multi-Modal Regularized Gaussian Reconstruction**: In addition to the photometric loss used in 3DGS, we incorporate depth, normal, and virtual-view constraints to regularize the optimization process (see Section IV-C). 3) **Iterative Gaussian Refinement**: We use image diffusion priors to enhance rendered images from 3DGS. These improved images further refine 3DGS optimization iteratively, incorporating diffusion model priors to boost detail and quality in novel view synthesis (see Section IV-D). 4) **Scene Enhancement**: In addition to image diffusion priors, we apply video diffusion priors to further enhance the rendered images from 3DGS, enhancing the realism of visual effects (refer to Section IV-E).

B. Background-Aware Depth-guided Initialization

Traditionally, 3DGS relies on point clouds and camera poses calculated through Structure from Motion (SfM) methods for initialization. However, SfM methods often encounter challenges in sparse view settings. To address this issue, we propose leveraging stereo priors [73] as a solution. DUSt3R, an end-to-end dense stereo model, can take sparse views as input and produce dense point clouds along with camera poses. Nevertheless, the point clouds generated by DUSt3R are prone to issues such as floating objects, artifacts, and distortion, particularly in the background of the 3D scene.

To overcome these challenges, we introduce the Background-Aware Depth-guided Initialization module to generate dense and precise point clouds. This module

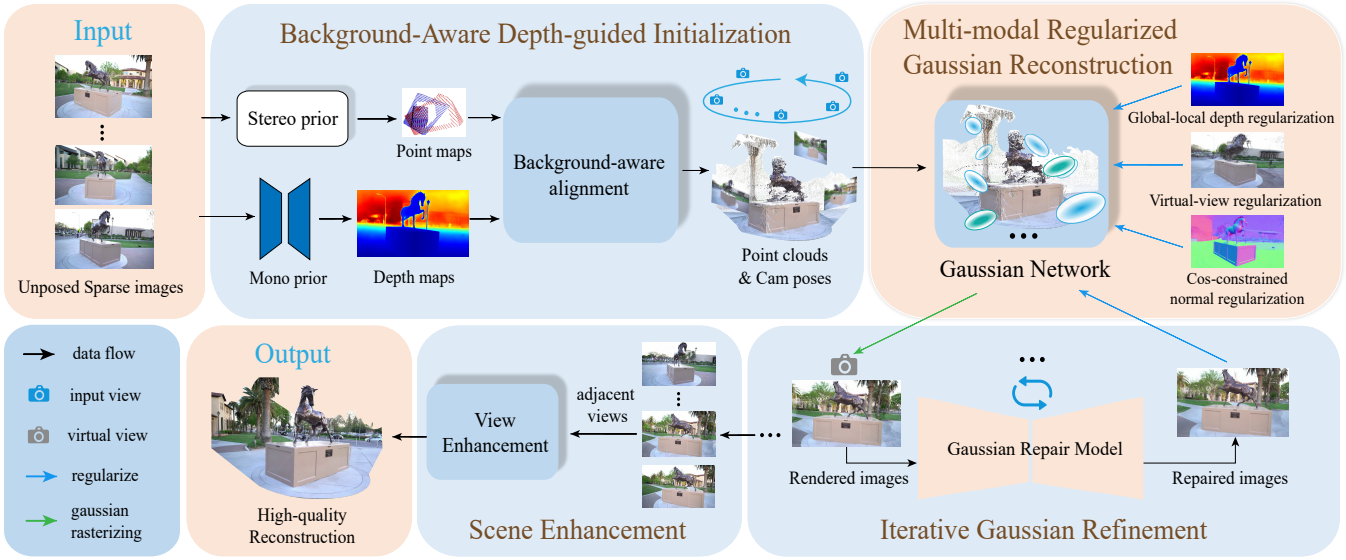


Fig. 1. **The Framework of LM-Gaussian.** Our method takes unposed sparse images as inputs. For example, we select 8 images from the Horse Scene to cover a 360-degree view. Initially, we utilize a Background-Aware Depth-guided Initialization Module to generate dense point clouds and camera poses (see Section IV-B). These variables act as the initialization for the Gaussian kernels. Subsequently, in the Multi-modal Regularized Gaussian Reconstruction Module (see Section IV-C), we collectively optimize the Gaussian network through depth, normal, and virtual-view regularizations. After this stage, we train a Gaussian Repair model capable of enhancing Gaussian-rendered new view images. These improved images serve as guides for the training network, iteratively restoring Gaussian details (see Section IV-D). Finally, we employ a scene enhancement module to further enhance the rendered images for realistic visual effects (see Section IV-E).

incorporates four key techniques: 1) **Camera Pose Recovery**: Initially, sparse images are used to generate point clouds for each image using DUSt3R. Subsequently, the camera poses and point clouds are aligned into a globally consistent coordinate system. 2) **Depth-guided Optimization**: Depth-guided optimization is then employed to refine the aligned point cloud. In this step, a monocular estimation model is used as guidance for the optimization process. 3) **Point Cloud Cleaning**: Two strategies are implemented for point cloud cleaning: geometry-based cleaning and confidence-based cleaning. During optimization, after every ξ iterations, a geometry-based cleaning step is executed to remove unreliable floaters. Following the optimization process, confidence-based cleaning is applied to distinguish between foreground and background, utilizing specific filtering techniques to preserve the final output point cloud. Next, we will provide detailed insights into the implementation of each component within this module.

Camera pose recovery: We first use minimum spanning tree algorithm [32] to align all camera poses and point clouds into a unified coordinate system. An optimization scheme is then utilized to enhance the quality of the aligned point clouds. Initially, following the approach of DUSt3R, a point cloud projection loss $\mathcal{L}_{pc}$ is minimized. Consider the image pair $\{\boldsymbol{I}_k, \boldsymbol{I}_l\}$ where $\boldsymbol{P}_k$ and $\boldsymbol{P}_l$ denote the point map in the k_{th} and l_{th} camera's coordinate system. The objective is to evaluate the consistency of 3D points in the k_{th} coordinate system with those in the l_{th} coordinate system. The projection loss is computed by projecting the point map $\boldsymbol{P}_l$ to the k_{th} coordinate system using a transformation matrix $\boldsymbol{T}_{k,l}$ that converts from the l_{th} coordinate system to the k_{th} coordinate system. The loss parameters include the transformation matrix $\boldsymbol{T}_{k,l}$, a scaling factor $\sigma_{k,l}$, and $\boldsymbol{P}_k$. This process is repeated for the remaining image pairs.

$$\mathcal{L}_{pc} = \sum_{k \in K} \sum_{l \in K \setminus \{k\}} \boldsymbol{\eta}_k \cdot \boldsymbol{\eta}_l \left\| \boldsymbol{P}_k - \sigma_{k,l} \boldsymbol{T}_{k,l} \boldsymbol{P}_l \right\| \tag{6}$$

The purpose of this loss function is to systematically pair each input image like $\boldsymbol{I}_k$ with all other images such as $\boldsymbol{I}_l$. For the image pair $\{\boldsymbol{I}_k, \boldsymbol{I}_l\}$, the loss function measures the disparity between the point map $\boldsymbol{P}_k$ in the k_{th} coordinate system and the transformed point map $\sigma_{k,l}\boldsymbol{T}_{k,l}\boldsymbol{P}_l$. These comparisons are weighted by their respective confidence maps $\boldsymbol{\eta}_k$ and $\boldsymbol{\eta}_l$.

Depth-guided Optimization: The optimization based solely on the projection loss may not be sufficient for reconstructing large-scale scenes, as it could lead to issues like floaters and scene distortion that can impact subsequent reconstructions. To tackle scene distortion, we integrate a robust model prior to guide the optimization network. Marigold, a diffusion-based monocular depth estimation model known for its top-tier performance in this domain, is employed to provide insights into the scene's depth information.

The monocular depth estimation model significantly enhances depth perception across different scales. Its guidance is pivotal in mitigating distortion issues and improving overall scene depth perception. Within the optimization network, we merge DUSt3R outputs with depth guidance by incorporating a point cloud projection loss, a multi-scale depth loss and a depth smoothness loss.

$$\mathcal{L}_{opt} = \mathcal{L}_{pc} + \alpha_d \mathcal{L}_D + \alpha_s \mathcal{L}_{smooth} \tag{7}$$

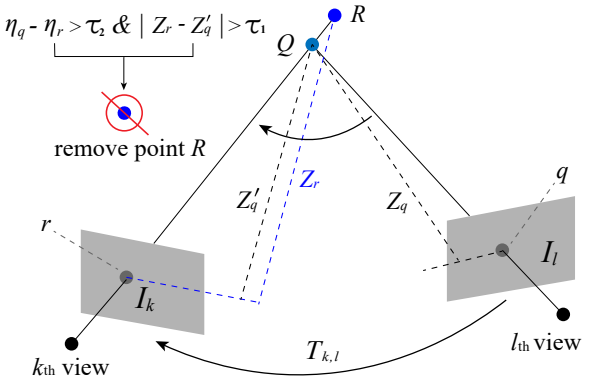


Fig. 2. **The depicted curve illustrates the point cleaning operation.** We project 3D point Q from the l_{th} coordinate system into the k_{th} coordinate system. If the difference between the projected depth Z'_q and the reference depth Z_r exceeds a threshold τ_1, and the confidence η_q of point Q exceeds the confidence η_r of point R by more than τ_2, we classify point R as artifacts and proceed to remove them. Otherwise we keep both two points.

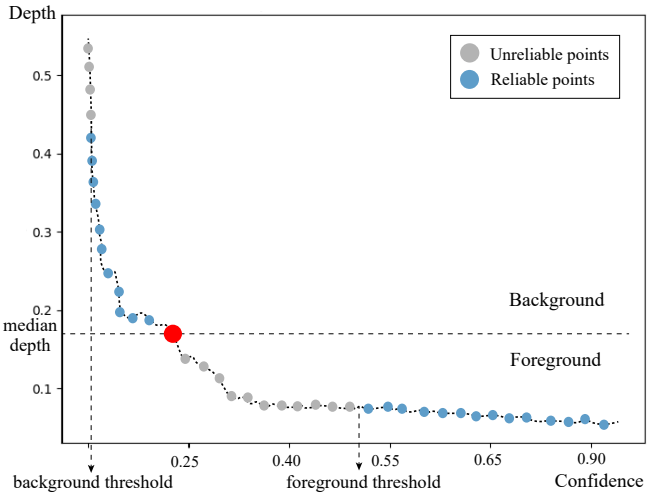


Fig. 3. The diagram depicted showcases the relationship between depth values and confidence of the 3D points. Points with a large distance exhibit decreased reliability, due to the bias of DUSt3R. To address this problem, we first divide the points into foreground and background parts based on median depth. Instead of using a single confidence threshold for the whole scene, we use two separate confidence thresholds to process the foreground and background parts individually.

where $\mathcal{L}_{opt}$ refers to the total optimization loss. α_d and α_s are loss weights of multi-scale depth loss $\mathcal{L}_D$ and depth smoothness loss $\mathcal{L}_{smooth}$. The smoothness loss encourages depth map smoothness by penalizing depth gradient changes, weighted by the image gradients and details of the multi-scale depth loss term $\mathcal{L}_D$ would be discussed later (see Sec IV-C).

Point cloud Cleaning: In order to eliminate floaters and artifacts, we implement two strategies for cleaning the point cloud: geometry-based cleaning and confidence-based cleaning.

In **geometry-based cleaning**, we adopt an iterative approach to remove unreliable points during the depth optimization process. For a set of K input images, as illustrated in Figure 2, the method involves systematically pairing image $\boldsymbol{I}_k$ with all other $K-1$ images within a single iteration. For the image pair $\boldsymbol{I}_k, \boldsymbol{I}_l$, a pixel q in $\boldsymbol{I}_l$ corresponds to a 3D point Q in the scene, represented as (X_q, Y_q, Z_q) in the l_{th} coordinate system. This point can be translated into the k_{th} coordinate system using the transformation matrix $\boldsymbol{T}_{k,l}$. The projected point intersects with $\boldsymbol{I}_k$ at pixel r, and its depth in the k_{th} coordinate system is denoted as Z'_q. Conversely, pixel r also corresponds to another 3D point R, denoted as (X_r, Y_r, Z_r) in the k_{th} coordinate system.

To tackle the floating issue, if we detect that the difference between the projected depth Z'_q and the depth Z_r of point R exceeds a threshold τ_1, and the confidence η_q of point Q exceeds the confidence η_r of point R by more than τ_2, we label point R as unreliable. As a result, we remove this point from the set of 3D points.

$$\left|Z_r - Z'_q\right| > \tau_1 \text{ and } \quad \eta_q - \eta_r > \tau_2 \Rightarrow \text{Exclude point } R \quad (8)$$

The cleaning operation of the point clouds is executed once every ξ iterations, with τ_1 and τ_2 serving as hyperparameters.

In addition to the geometry-based cleaning process, we also implement a **confidence-based cleaning** step post-optimization. Each point within the point clouds is assigned a confidence value. The original DUSt3R method applies a basic confidence threshold to filter out points with confidence below a certain level. However, due to the distance bias, this approach may inadvertently exclude many background elements. To tackle this challenge, as depicted in Figure 3, we differentiate between foreground and background regions by arranging the depths of all points and selecting the median depth as the separation boundary. Points in the foreground, typically observed in multiple-view images, tend to have higher confidence levels. Consequently, we establish a high-confidence threshold for foreground objects. On the contrary, the background area, often captured from a distance and present in only a few images, tends to exhibit lower confidence levels. Hence, we adopt a more lenient strategy for this region, employing a lower confidence threshold for point cleaning.

C. Multi-modal Regularized Gaussian Reconstruction

Dense point clouds and camera poses are acquired through Background-Aware Depth-guided Initialization. These variables serve as the initialization of Gaussian kernels. Vanilla 3DGS methods utilize photo-metric loss functions such as $\mathcal{L}_1$ and $\mathcal{L}_{SSIM}$ to optimize 3DGS kernels and enable them to capture the underlying scene geometry. However, challenges arise in scenarios with extremely sparse input images. Due to the inherent biases of the Gaussian representation, the Gaussian kernels are prone to overfitting on the training views and cause degradation on unseen perspectives. To mitigate this issue, we enhance the Gaussian optimization process by integrating photo-metric loss, multi-scale depth loss, cosine-constrained normal loss, and norm-weighted virtual-view loss.

Photo-metric Loss: In line with vanilla 3DGS, we initially compute the photo-metric loss between the input RGB images

and Gaussian-rendered images. The photo-metric loss function combines $\mathcal{L}_1$ with an SSIM term $\mathcal{L}_{SSIM}$.

$$\mathcal{L}_{pho} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{SSIM} \tag{9}$$

where λ represents a hyperparameter, and $\mathcal{L}_{pho}$ denotes the photo-metric loss.

Multi-scale Depth Regularization: To address above challenges, depth information can be incorporated into the Gaussian scene to provide regularization during training. Similar to the Initialization Module, we initially employ the monocular estimation model Marigold to predict depth images $\{\hat{\boldsymbol{D}}_k\}_{k=0}^{K-1}$ from sparse input images. Since monocular depth estimation models typically offer relative depth predictions without realistic scene scale information, we utilize the Pearson Correlation Coefficient (PCC) [14] as a metric to gauge the similarity between depth maps. The Pearson Correlation Coefficient is a fundamental statistical correlation coefficient that quantifies the linear correlation between two data sets. Essentially, it assesses the resemblance between two distinct distributions X and Y.

$$\mathrm{PCC}(X,Y) = \frac{E[XY]-E[X]E[Y]}{\sqrt{E[Y^2]-E[Y]^2}\sqrt{E[X^2]-E[X]^2}} \tag{10}$$

where E represents the mathematical expectation.

Inspired by previous works [34] [81], to enhance the capture of local structures, we go beyond assessing the correlation between depth maps at the source scales. We divide depth images into small patches and compare the correlation among these depth patches. The Pearson correlation coefficient has a strong connection with normalized cross-correlation, implying that the utilization of this loss function promotes high cross-correlation values for patches at corresponding locations in both depth maps, regardless of depth value variations. During each iteration, we randomly select F non-overlapping patches to evaluate the depth correlation loss, defined as:

$$\mathcal{L}_{depth} = \frac{1}{F}\sum_{f=0}^{F-1} 1 - \mathrm{PCC}(\bar{\mathbf{\Gamma}}_f, \hat{\mathbf{\Gamma}}_f) \tag{11}$$

where $\bar{\mathbf{\Gamma}}_f$ denotes the f_{th} patch of Gaussian-rendered depth maps and $\hat{\mathbf{\Gamma}}_f$ denotes the f_{th} patch of depth maps predicted by monocular estimation model. Intuitively, this loss works to align depth maps of the Gaussian representation with the depth map of monocular prediction, mitigating issues related to inconsistent scale and shift.

Cosine-constrained Normal Regularization: While depth provides distance information within the scene, normals are also essential for shaping surfaces and ensuring smoothness. Therefore, we introduce a normal-prior regularization to constrain the training process.

We utilize cosine similarity to quantify the variance between the predicted normal maps got from normal prior [28] and the normal maps rendered using Gaussian representations.

$$\mathcal{L}_{normal} = \frac{1}{K}\sum_{k=0}^{K-1} 1 - \mathrm{COS}(\bar{\boldsymbol{N}}_k, \hat{\boldsymbol{N}}_k) \tag{12}$$

where $\bar{\boldsymbol{N}}_k \in \mathbb{R}^{H\times W\times 3}$ represents the Gaussian-rendered normal maps, and $\hat{\boldsymbol{N}}_k$ signifies the normal maps predicted by the monocular estimation model. The function $\mathrm{COS}()$ denotes the cosine similarity function.

Weighted Virtual-view Regularization: In cases where the training views are sparse, the Gaussian scene may deteriorate when presented with new views due to the lack of supervision from these training views. Hence, we introduce a virtual-view regularization strategy to preserve the original point cloud information throughout the optimization process.

As illustrated in Figure 4, we randomly sample K_v virtual views in 3D space. For each virtual camera, we project the point clouds onto the 2D plane of the view. A weighted blending algorithm is employed to render the 3D points into RGB point-rendered images. These point-rendered images serve as guidance for the Gaussian optimization process.

When creating a point-rendered RGB image from a virtual view, the color of each pixel i is determined by the U nearest projected 3D points. As shown in Figure 5, these points are selected based on their proximity to pixel i within a radius π, where π is defined as one-third of the pixel width. Subsequently, these selected points are arranged in order of their distances $\{d_u\}_{u=0}^{U-1}$ from the viewpoint. Weights are then allocated to these ordered points according to their distances, with closer 3D points to the virtual viewpoint receiving higher weights.

$$c(i) = \begin{cases} \sum_{u=0}^{U-1} c_u w_u, \quad w_u = \frac{e^{-d_u}}{\sum_{u=0}^{U-1} e^{-d_u}} & \text{if valid points} \\ c_{bg} & \text{otherwise} \end{cases}$$

Here, $c(i)$ represents the color of pixel i after point rasterization. c_u denotes the color of the u_{th} 3D point, and w_u is its corresponding weight. In cases where a pixel has no valid point projection (i.e., $U=0$), we assign the pixel the predefined background color c_{bg}, which, in this instance, is white.

By employing the norm-weighted blending algorithm, we obtain K_v point-rendered RGB images denoted as $\{\boldsymbol{I}_k^{pr}\}_{k=0}^{K_v-1}$. These images are subsequently utilized to regulate Gaussian kernels, thereby imposing constraints on optimization and preventing overfitting. The virtual-view loss function at this stage is presented below.

$$\begin{aligned}\mathcal{L}_{vir} = (1-\lambda)\mathcal{L}_1(\bar{\boldsymbol{I}}_k, \boldsymbol{I}_k^{pr}) + \\ \lambda\mathcal{L}_{SSIM}(\bar{\boldsymbol{I}}_k, \boldsymbol{I}_k^{pr}), k \in K_v\end{aligned} \tag{13}$$

where λ, $\mathcal{L}_1$, $\mathcal{L}_{SSIM}$ are the same as original 3d Gaussian splatting and $\bar{\boldsymbol{I}}_k$ is the Gaussian-rendered RGB image from k_{th} view.

Multi-modal Joint Optimization: Throughout the Multi-modal Regularized Gaussian Reconstruction phase, in addition to the photo-metric loss, Multi-scale Depth Regularization, Cosine-constrained Normal Regularization, and Norm-weighted Virtual-view Regularization are incorporated to steer the training process. These methodologies are pivotal in alleviating overfitting and upholding the output quality.

$$\mathcal{L}_{multi} = \mathcal{L}_{pho} + \beta_{vir}\mathcal{L}_{vir} + \beta_{dep}\mathcal{L}_{\text{depth}} + \beta_{nor}\mathcal{L}_{\text{normal}} \tag{14}$$

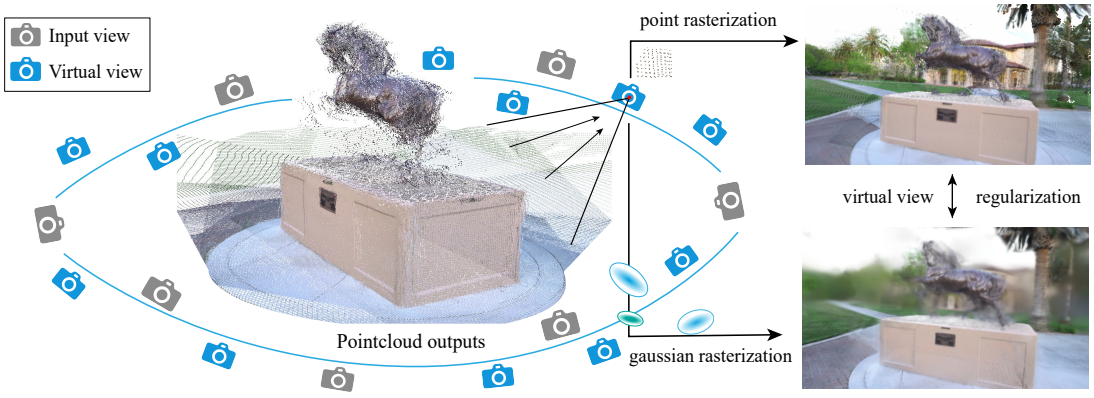


Fig. 4. **Visual Process of Weighted Virtual View Regularization.** For each virtual view, we employ two distinct methods for image rasterization. First, we utilize Gaussian kernels to produce a Gaussian-rendered image. Then, we apply a weighted blending algorithm to create a point-rendered image. We enforce consistency between these two images.

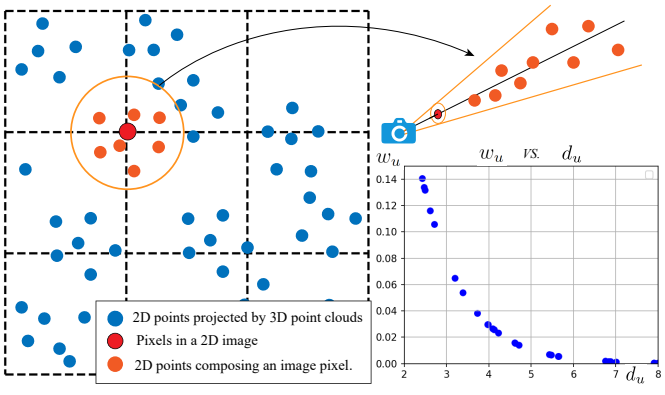


Fig. 5. **Illustration of Weighted Blending for an Image Pixel.** The blue point represents the 2D projection of a point from the 3D point clouds. The red point corresponds to a pixel on the RGB image. The orange points indicate the selected points that contribute to the final color of the red pixel. The scatter plot demonstrates the diverse weights assigned to 3D points at different depths, with U fixed at 30.

where $\mathcal{L}_{multi}$ represents the loss function utilized in the Multi-modal Regularized Gaussian Reconstruction. The weights $\beta_{vir}, \beta_{dep}, \beta_{nor}$ serve as hyperparameters to regulate their impact, with further elaboration provided in the Experiment Section.

D. Iterative Gaussian Refinement

During this phase, we implement an iterative optimization approach to progressively enhance scene details. Initially, we uniformly enhance the Gaussian-rendered images from virtual viewpoints using a Gaussian repair model. This model refines blurry Gaussian-rendered images into sharp, realistic representations. Following this enhancement, these refined images act as supplementary guidance, facilitating the optimization of Gaussian kernels in conjunction with depth and normal regularization terms. After ζ optimization steps, we re-render the Gaussian images and subject them to the repair model once more, replacing the previously refined images for another iteration of supervision.

*1) **Iterative Gaussian Optimization**:* Initially, we generate K_v images using Gaussian kernels from virtual viewpoints. Subsequently, the Gaussian Repair Model is utilized to uniformly enhance these images, resulting in a set of repaired images denoted as $\{\boldsymbol{I}_k^{repair}\}_{k=0}^{K_v-1}$. To maintain scene coherence and reduce potential conflicts, we set the denoise strength to a low value and gradually reintroduce limited details to the Gaussian-rendered images during each repair process. These repaired virtual-view images, along with monocular depth and normal maps from training views as outlined in Section IV-C, are then employed to regulate the Gaussian optimization. The overall optimization loss in the Gaussian refinement stage, denoted as $\mathcal{L}_{refine}$, is determined by:

$$\mathcal{L}_{refine} = \mathcal{L}_{pho} + \beta_{rep}\mathcal{L}_{rep} + \beta_{dep}\mathcal{L}_{depth} + \beta_{nor}\mathcal{L}_{normal} \quad (15)$$

Here, $\mathcal{L}_{depth}$ and $\mathcal{L}_{normal}$ correspond to the Multi-modal Regularized Gaussian Reconstruction. β_{rep} signifies the weight of the repair loss. The repair loss $\mathcal{L}_{rep}$ is defined as follows:

$$\begin{aligned}\mathcal{L}_{rep} = (1-\lambda)\mathcal{L}_1(\bar{\boldsymbol{I}}_k, \boldsymbol{I}_k^{repair}) + \\ \lambda\mathcal{L}_{SSIM}(\bar{\boldsymbol{I}}_k, \boldsymbol{I}_k^{repair}), k \in K_v\end{aligned} \quad (16)$$

In this formulation, we leverage the photo-metric loss between the repaired images $\{\boldsymbol{I}_k^{repair}\}_{k=0}^{K_v-1}$ and the Gaussian-rendered images $\{\bar{\boldsymbol{I}}_k\}_{k=0}^{K_v-1}$ in one loop, utilizing the repaired images as a guiding reference. The parameters λ, $\mathcal{L}_1$, and $\mathcal{L}_{SSIM}$ remain consistent with the original 3D Gaussian splatting methodology. The above operations will be repeated.

Through this iterative optimization strategy, the newly generated images gradually enhance in sharpness without being affected by blurring caused by view disparities. The optimization process persists until it is ascertained that the diffusion process no longer produces satisfactory outcomes, as evidenced by deviations from the initial scene or inconsistencies across various viewpoints.

*2) **Gaussian Repair Model**:* In this section, we present the Gaussian repair model utilized earlier. Its primary objective is to enhance blurry Gaussian-rendered images into sharp, realistic images while preserving the style and content of the original image.

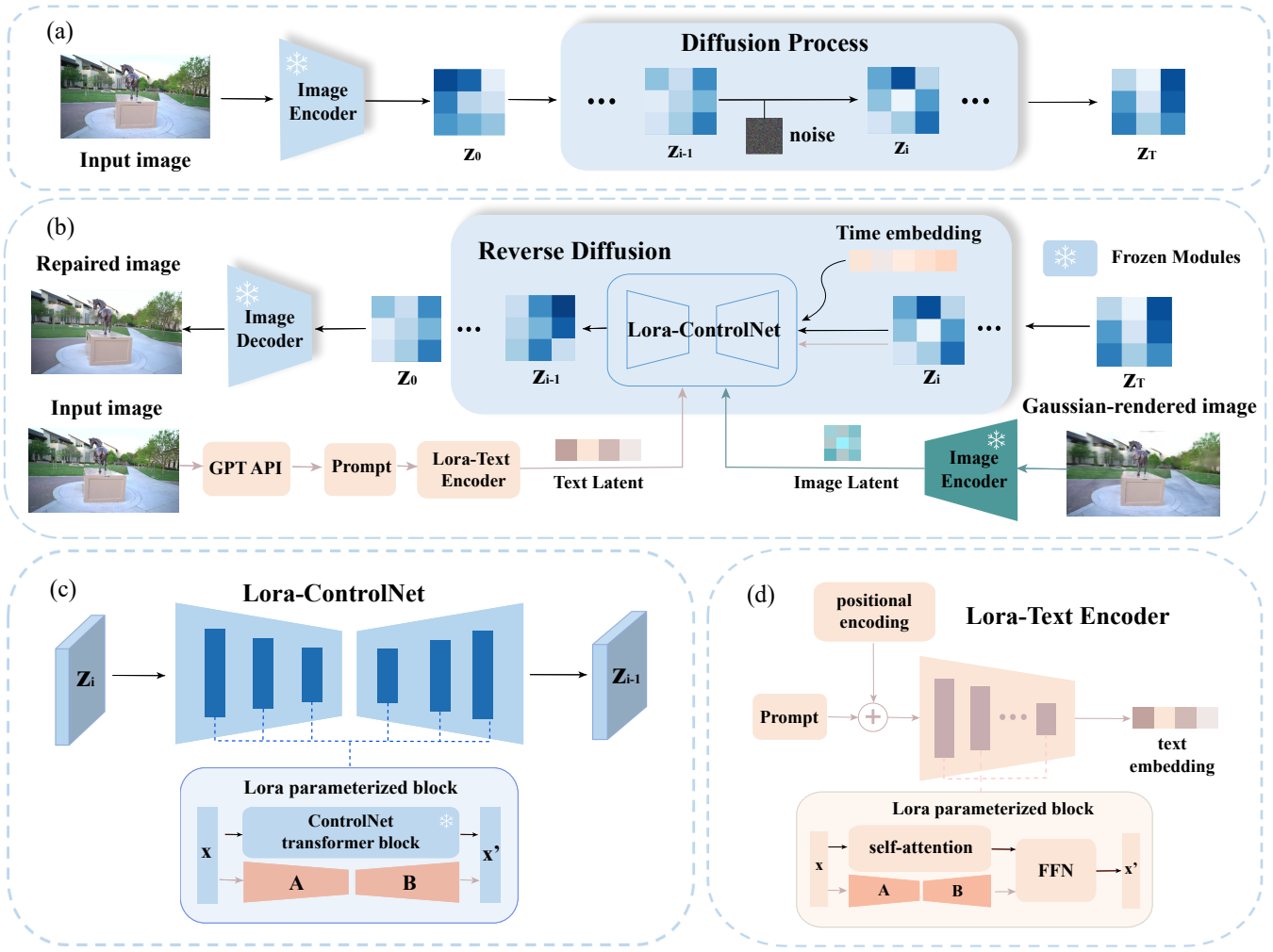


Fig. 6. **Architecture of Our Gaussian Repair Model.** The architectural layout of our Gaussian Repair Model entails selectively freezing all parameters except for the Lora weight within the network to fine-tune a ControlNet. This process imbues scene characteristics into the model, augmenting its ability to rectify blurry scene images. Precisely, we integrate Lora parameters into the text condition encoder and ControlNet's UNet. With a Lora rank designated as 16, this integration takes place in each transformer block, linear layer, and convolutional layer. By training on image pairs from the coarse stage, our model excels in generating detailed real-world images from blurry Gaussian-rendered counterparts, providing guidance for subsequent optimization.

Model Architecture: The architecture of the Gaussian Repair Model is illustrated in Figure 6. It takes Gaussian-rendered images and real-world input images as inputs. In Figure 6(b), the Gaussian-rendered image $\bar{\boldsymbol{I}}$ undergoes image encoding to extract latent image features. The real-world input images are processed through a GPT API for a description prompt σ, then encoded to obtain text latent features. These image and text latent features act as conditions for a ControlNet [96] to predict noise ϵ_θ and progressively remove noise from the Gaussian-rendered image. The model is a controlnet finetuned by injecting lora into its layer and it can produce the repaired Gaussian-rendered image. Figure 6(c) provides insight into the Lora-ControlNet, where Lora [20] weights are integrated into each transformer layer of the ControlNet's UNet. We maintain the original parameters of the transformer blocks constant and solely train the low-rank compositions A, B, where $A \in \mathbb{R}^{d\times r}, B \in \mathbb{R}^{r\times k}$, with a rank $r \ll \min(d, k)$. Concerning the text encoder, as depicted in Figure 6(d), Lora weights are integrated into each self-attention layer of the encoder. The input of the Lora-Text Encoder is the scene prompt, and the output is the text embedding.

Training process: In this section, we will explore the training process of the Gaussian repair model. Initially, for data preparation, we collect image pairs from Section IV-C, where input images within each scene act as reference images. For each training perspective, we randomly select ω Gaussian-rendered images and pair them with input images to create training pairs. Subsequently, these image pairs are utilized to train our Gaussian Repair Model. As shown in Figure 6(a), the input image $\boldsymbol{I}$ undergoes a forward diffusion process. Specifically, the image is input into an image encoder to extract the latent representation z_1. This latent representation then undergoes a diffusion process where noise ϵ is gradually introduced over T steps. After obtaining latent z_T, a reverse diffusion process is initiated, as illustrated in Figure 6(b), where a Lora-UNet and a Lora-ControlNet are employed to predict noise ϵ_θ at each step. This predicted noise, combined with the noise introduced

TABLE II
QUANTITATIVE ANALYSIS ON TANKS AND TEMPLES DATASETS WITH 8 INPUT VIEWS. OUR APPROACH PRODUCES SUPERIOR RESULTS AND FINER DETAILS, AS INDICATED BY PSNR, SSIM, AND LPIPS METRICS, OUTPERFORMING BASELINE METHODS SUCH AS DNGAUSSIAN, FREENERF, AND SPARSENERF.

Methods	Ours			DNGaussian [34]			FreeNerf [86]			SparseNerf [69]		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
Family	0.63	18.46	0.23	0.34	12.98	0.57	0.28	11.42	0.54	0.32	11.94	0.53
Horse	0.70	18.97	0.18	0.37	13.77	0.53	0.31	11.91	0.49	0.35	12.68	0.47
Ignatius	0.52	17.94	0.28	0.26	11.88	0.66	0.25	10.64	0.56	0.28	10.54	0.59
Trunk	0.62	18.24	0.27	0.31	12.03	0.62	0.27	10.99	0.58	0.26	10.67	0.61
Avg.	**0.62**	**18.40**	**0.24**	0.32	12.67	0.59	0.28	11.24	0.54	0.30	11.45	0.57

during the diffusion process, contributes to calculating the loss function, aiding in the training process. The loss function for any input image can be defined as follows.

$$\mathcal{L}_{Control} = E_{\boldsymbol{I},t,\bar{\boldsymbol{I}},\epsilon\in\mathcal{N}(0,1)}[\|(\epsilon_\theta(\boldsymbol{I},t,\bar{\boldsymbol{I}},\sigma)-\epsilon)\|_2^2] \quad (17)$$

Here, σ refers to the text prompt of the reconstructed scene. $\boldsymbol{I}$ represents the actual input image, and $\bar{\boldsymbol{I}}$ is the Gaussian-rendered image obtained from Section IV-C. $E_{\boldsymbol{I},t,\bar{\boldsymbol{I}},\epsilon\in\mathcal{N}(0,1)}$ denotes the expectation over the input image $\boldsymbol{I}$, the time step t, the text prompt σ, the condition image $\bar{\boldsymbol{I}}$, and the noise ϵ drawn from a normal distribution with mean 0 and standard deviation 1. ϵ_θ indicates the predicted noise.

E. Scene Enhancement

Given the sparse input images and the restricted training perspectives, it is expected that rendered images from adjacent new viewpoints may display discrepancies. In order to ensure high-quality and consistent rendering along a specified camera path, we propose a View Enhancement module, which utilizes video diffusion priors to improve the coherence of rendered images.

This module concentrates on enhancing the visual consistency of rendered images without delving into Gaussian kernel refinement. Initially, multiple images are rendered along a predetermined camera trajectory and grouped for processing. Subsequently, a video diffusion UNet is employed to denoise these images to generate enhanced images. In the video diffusion model, DDIM inversion [64] is utilized to map Gaussian-rendered images back to the latent space, the formulation can be expressed as:

$$z_{t+1} = \frac{\overline{\alpha}_{t+1}}{\overline{\alpha}_t} z_t + (\frac{1}{\overline{\alpha}_{t+1}} - 1 - \frac{1}{\overline{\alpha}_t} - 1)\epsilon_\theta(z_t, t, \sigma), \quad (18)$$

where $t \in [1, T]$ is the time step and $\overline{\alpha}_t$ denotes a decreasing sequence that guides the diffusion process. σ serves as an intermediate representation that encapsulates the textual condition.

The rationale behind mapping Gaussian-rendered images to a latent space is to leverage the continuous nature of the latent space, preserving relationships between different views. By denoising images collectively in the latent space, the aim is to enhance visual quality without sacrificing spatial consistency. In the scene enhancement model, we utilized Zeroscope-XL as the video-diffusion prior and set the denoising strength to 0.1.

V. EXPERIMENTS

A. Experimental Setup

Dataset: Our experiments were conducted using three datasets: the Tanks and Temples Dataset [31], the MipNeRF360 Dataset [4], and the LLFF Dataset [50]. The Tanks and Temples and MipNeRF360 datasets feature 360-degree real-world scenes, while the LLFF dataset comprises feed-forward scenes. From the Tanks and Temples Dataset, we uniformly selected 200 images covering scenes like Family, Horse, Ignatius, and Trunk to represent the entire 360-degree environments. In the MipNeRF360 dataset, we chose the initial 48 frames capturing various elevations across a full 360-degree rotation, including scenes such as Garden, Bicycle, Bonsai, and Stump. Additionally, scenes like flowers, orchids, and ferns from the LLFF Dataset have been incorporated as well.

Train/Test Datasets Split: For 360-degree scenes, we varied the view number K from 4 to 16 to assess all the algorithms under consideration. In a training set comprising K images, the remaining images were allocated to the test set. Concerning the feed-forward dataset, we adhered to the setup outlined in previous research [34], [55], [69], employing 3 views for training and the remainder for testing.

Metrics: In the assessment of novel view synthesis, we present Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [74], and Learned Perceptual Image Patch Similarity (LPIPS) [97] scores as quantitative measures to evaluate the reconstruction performance.

Baselines: We compare our method against 7 baseline approaches. Our evaluation included sparse-view reconstruction methods such as DNGaussian [34], FreeNeRF [86], SparseNeRF [69], PixelNeRF [90], MVSNeRF [8], DietNeRF [26], and RegNerf [55]. Additionally, we compared our method with the vanilla 3DGS approach. Notably, all these baseline methods employed Colmap for the pre-computation of camera parameters.

B. Implementation Details

We implemented our entire framework in PyTorch 2.0.1 and conducted all experiments on an A6000 GPU. In the Background-Aware Depth-guided Initialization stage, loss weights α_g, α_s and α_l were set to 0.01, 0.01 and 0.1. Geometry-based cleaning was applied every 50 iterations. For Confidence-based cleaning, τ_1 and τ_2 were set to 0.1 and

Fig. 7. **Qualitative comparison on Tanks and Temples Dataset with 16 input views.** Our approach consistently fairs better in recovering image structure from foggy geometry, where baselines typically struggle with floaters and artifacts.

TABLE III
QUANTITATIVE EVALUATIONS ON MIPNERF360 DATASETS WITH 8 INPUT VIEWS.

Methods	Ours			DNGaussian [34]			FreeNerf [86]			SparseNerf [69]		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
Garden	0.58	17.70	0.33	0.37	12.74	0.62	0.31	11.82	0.58	0.33	12.04	0.59
Bicycle	0.51	16.85	0.40	0.33	12.26	0.70	0.29	11.62	0.67	0.30	11.64	0.68
Bonsai	0.61	17.25	0.29	0.41	12.37	0.67	0.33	11.46	0.65	0.32	11.60	0.69
Stump	0.49	17.84	0.30	0.30	12.48	0.63	0.25	11.76	0.61	0.27	11.84	0.60
Avg.	**0.55**	**17.49**	**0.33**	0.35	12.46	0.65	0.30	11.67	0.63	0.31	11.78	0.62

0, respectively. Moving on to the Multi-modal Regularized Gaussian Reconstruction stage, we trained the Gaussian model for 6,000 iterations with specified loss function weights: β_{vir} = 0.5, β_{dep} = 0.3, and β_{nor} = 0.1 across all experiments. We use ControlNet [96] as our foundational model. The low-rank r was set to 64, and the model was trained for 2000 steps. In the Iterative Gaussian Refinement Module, the denoising strength was set at 0.3 for each repair iteration, and the repair process was repeated every 4,000 iterations. The value of β_{rep} was adjusted based on the distance between the virtual view and its nearest training view, within the range of (0, 1). This iterative cycle was set to 3 in our experiments.

C. Quantative and Qualitative Results

Tanks and Temples & MipNerf360: The quantitative results, presented in Table II and Table III, demonstrate that our method outperforms others in terms of PSNR, SSIM, and LPIPS metrics, showcasing superior performance and finer details. Visual results are also showcased in Figure 7 and Figure 8. With 16 input images, LM-Gaussian produces high-quality reconstruction results, preserving most structures and details, whereas DNGaussian and SparseNerf yield cluttered outcomes. Despite DNGaussian and SparseNerf utilizing frequency and depth regularization to prevent overfitting, their performance falls short for two primary reasons. Firstly, their dependence on Colmap for initializing point clouds and camera poses proves challenging in sparse-view scenarios where traditional SfM methods struggle to provide reliable outputs. Secondly, these methods neglect to introduce significant model priors to restore intricate details. Given the sparse nature of the input data, critical details may be lost without additional information.

LLFF: Besides challenging 360-degree large-scale scenes, we also conducted experiments on feed-forward scenes like the LLFF Dataset to ensure the thoroughness of our study and validate the robustness of our method. Following previous

Fig. 8. **Qualitative comparison on MipNerf360 Dataset with 16 input views.** Similar to Tanks and Temple Dataset, our approach consistently fairs better in recovering image structure from foggy geometry, where baselines typically struggle with floaters and artifacts.

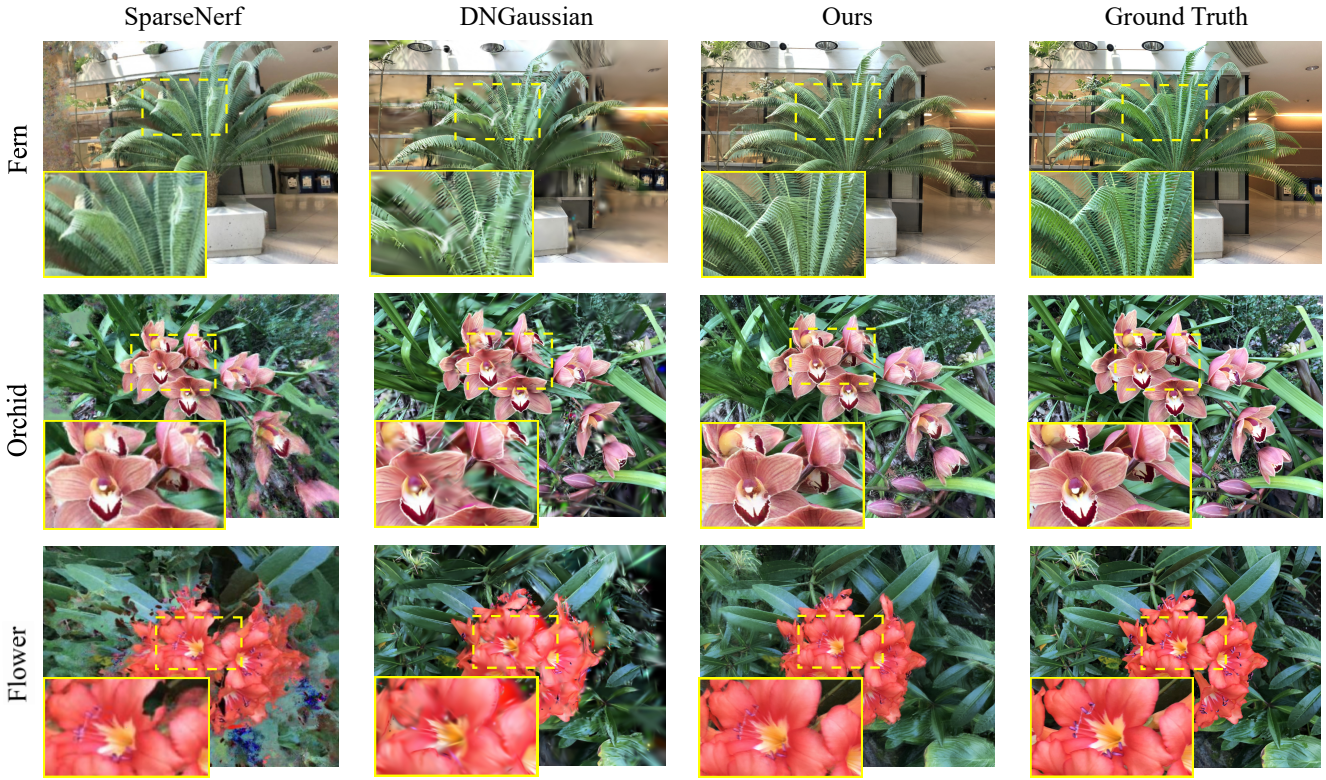
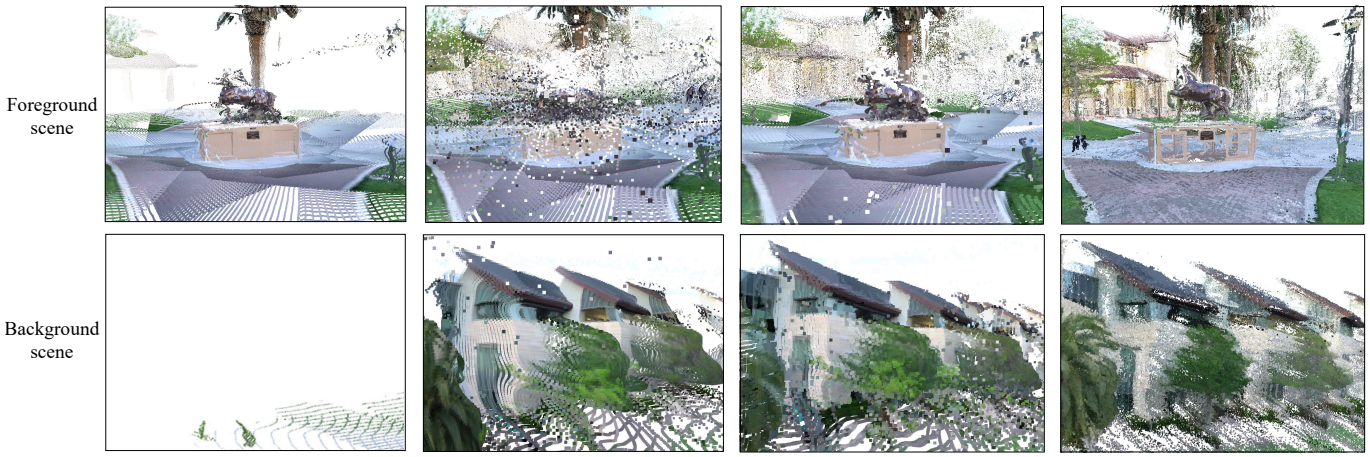


Fig. 9. **Qualitative comparison on LLFF Dataset with 3 input views.** Compared to baseline methods like DNGaussian, FreeNerf, and SparseNerf, our technique delivers enhanced results and greater detail, as demonstrated by PSNR, SSIM, and LPIPS scores.



(a) DUSt3R focusing on Foreground (8 images) (b) DUSt3R considering Background (8 images) (c) Background-Aware Depth-guided Initialization (8 images) (d) Dense Reconstruction from Colmap (200 images)

Fig. 10. **Point Cloud visualizations of DUSt3R and Background-Aware Depth-Guided Initialization.** As shown in images (a) and (b), with different confidence threshold for cleaning, DUSt3R either suffer from the empty or significant artifacts in the background part. Through the utilization of Depth-Guided Optimization, Point Cloud Cleaning, and Foreground-Background Separation techniques, our module excels in producing enhanced point clouds while addressing issues such as floaters and scene distortion. Image (d) displays the dense reconstruction result from 200 images by Colmap, serving as a useful reference.

TABLE IV
QUANTITATIVE COMPARISON ON LLFF DATASET FOR 3 INPUT VIEWS. THE BEST, SECOND-BEST, AND THIRD-BEST ENTRIES ARE MARKED IN RED, YELLOW, AND GREEN, RESPECTIVELY. OUR METHOD SHOWS THE BEST RECONSTRUCTION RESULT COMPARED WITH OTHER SPARSE-VIEW RECONSTRUCTION WORKS.

Methods	LLFF PSNR ↑	LLFF LPIPS ↓	LLFF SSIM ↑
PixelNeRF [90]	15.17	0.612	0.338
MVSNeRF [8]	16.88	0.427	0.484
DietNeRF [26]	14.94	0.496	0.370
RegNeRF [55]	18.08	0.396	0.487
FreeNeRF [86]	18.63	0.328	0.512
SparseNerf [69]	14.94	0.496	0.370
DNGaussian [34]	18.32	0.314	0.535
LM-Gaussian	**19.63**	**0.228**	**0.614**

sparse-view reconstruction works, we take 3 images as input, with quantitative results presented in Table IV and qualitative results shown in Figure 9. It is observed that sparse-view methods like DNGaussian also demonstrate commendable visual outcomes and relatively high PSNR and SSIM values. This can be attributed to the nature of the LLFF Dataset, which does not encompass a 360-degree scene but rather involves movement within a confined area. This results in higher image overlap and fewer unobserved areas, making it easier to reconstruct the scene. However, despite the impressive performance of these methods, our approach still exhibits certain advantages. In addition to the numerical enhancements demonstrated in TABLE IV, taking the flower scene as an example, our method excels in visual results by restoring finer details such as flower textures. Moreover, our method maintains superior performance in regions with less overlap, as exemplified by the leaves in the surroundings.

TABLE V
ABLATION STUDY OF DIFFERENT INITIALIZATIONS. WE COMPARE OUR BACKGROUND-AWARE DEPTH-GUIDED INITIALIZATION WITH COLMAP AND DUST3R ON THE HORSE SCENE WITH 16-VIEW SETTING.

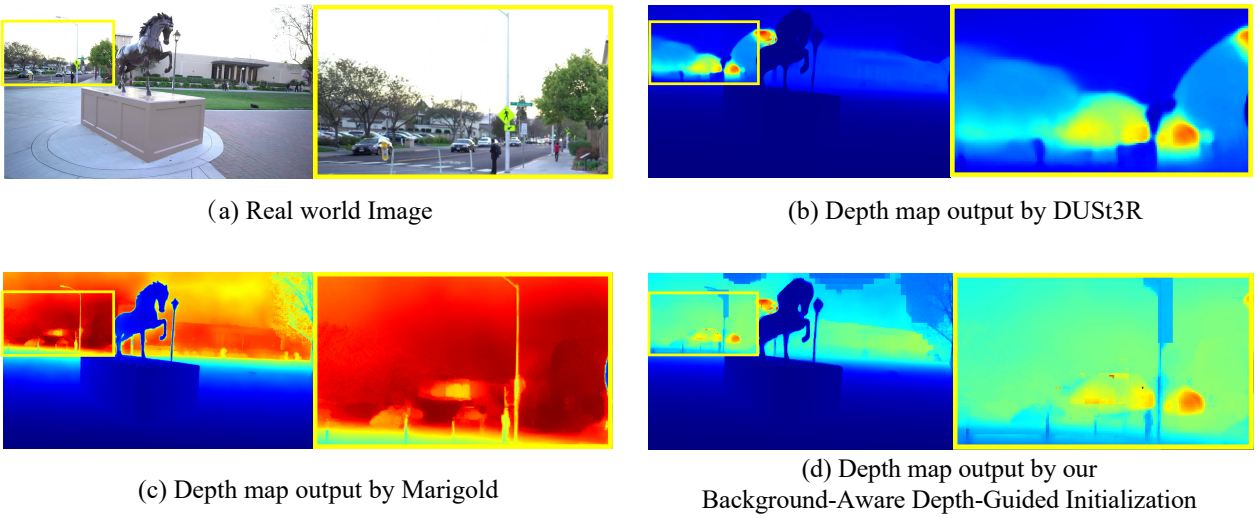
Initialization	PSNR ↑	LPIPS ↓	SSIM ↑
Colmap	13.42	0.558	0.192
DUSt3R	17.12	0.328	0.546
Proposed Initialization	**18.04**	**0.304**	**0.576**

D. Ablation Study

Colmap Initialization: Initially, we investigated the conventional Colmap method within our sparse-view settings. It fails to reconstruct point clouds with 8 input images in 360-degree scenes. We progressively increased the number of input images until Colmap could eventually generate sparse point clouds. With 16 input images, as illustrated in Figure 12, Colmap's resulting point clouds were significantly sparse, comprising only 1342 points throughout the scene. In contrast, our Background-Aware Depth-guided Initialization method excels in generating high-quality dense point clouds.

Effect of Background-Aware Depth-guided Initialization: Through visual demonstrations, we highlight DUSt3R's limitations in reconstructing high-quality background scenes, plagued by artifacts and distortions. As depicted in Figure 10(a)(b), DUSt3R either lacks background details or presents poor background quality, resulting in subpar reconstructions.

In contrast, as illustrated in Figure 10 (c), our module adeptly reconstructs background scenes with minimal distortion while preserving high-quality foreground scenes, exhibiting fewer artifacts and floaters compared to Figure (b)'s point clouds. Additionally, we present visualizations of depth maps



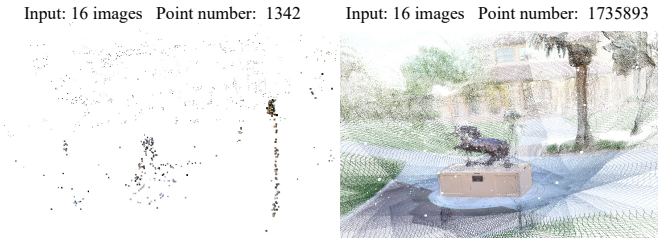
(a) Real world Image

(b) Depth map output by DUSt3R

(c) Depth map output by Marigold

(d) Depth map output by our Background-Aware Depth-Guided Initialization

Fig. 11. **Comparison of Point Cloud Depth Before and After Depth-Guided Optimization.** The depth maps generated by DUSt3R display a blurred background for the input image. In contrast, by leveraging depth priors, our Background-Aware Depth-Guided Initialization produces significantly enhanced depth maps, offering a more precise representation of the scene's depth. While the optimized depth may still exhibit some imperfections, particularly around the streetlights, these issues will be addressed in further refinement stages.



(a) Pointclouds output by Colmap

(b) Pointclouds output by Background-Aware Depth-guided Initialization

Fig. 12. Point clouds output by Colmap and Background-Aware Depth-guided Initialization. (a) Colmap fail to reconstruct reliable 3d points. (b) Dense pointclouds are obatined by Multi-modal Prior-guided Initialization

TABLE VI
ABLATION STUDY. WE ABLATE OUR METHOD ON THE HORSE SCENE WITH 16-VIEW SETTING. BACKGROUND-AWARE DEPTH-GUIDED INITIALIZATION(BA), REGULARIZATION STRATEGIES AND ITERATIVE GAUSSIAN REFINEMENT ALL IMPROVES THE NOVEL VIEW SYNTHESIS QUALITY.

Method			Metric		
BA	Regularization	Refinement	PSNR ↑	LPIPS ↓	SSIM ↑
×	×	×	13.42	0.558	0.192
✓	×	×	18.04	0.304	0.576
✓	✓	×	21.32	0.145	0.731
✓	✓	✓	**22.04**	**0.119**	**0.776**

TABLE VII
REGULARIZATION TEST. WE INDIVIDUALLY TEST THE MULTI-DEPTH REGULARIZATION, COSINE-NORMAL REGULARIZATION AND WEIGHTED POINT-RENDER REGULARIZATION

Regularizations			Metric		
Depth	Normal	Virtual-view	PSNR ↑	LPIPS ↓	SSIM ↑
×	×	×	18.04	0.304	0.576
✓	×	×	19.74	0.205	0.634
✓	✓	×	20.02	0.188	0.665
✓	✓	✓	**21.32**	**0.145**	**0.731**

before and after depth optimization. In Figure 11, the original DUSt3R's depth maps reveal background blurriness, blending elements like the sky and street lamps. With guidance from Marigold, our Background-Aware Depth-guided Initialization notably enhances the reconstruction of background scenes compared to the original DUSt3R output.

Moreover, Table VII provides quantitative comparisons among our method, Colmap, and DUSt3R. While Colmap yields less favorable results, DUSt3R shows significant improvement. Despite DUSt3R's performance, our initialization module achieves state-of-the-art outcomes, leading to improvements in PSNR and SSIM metrics.

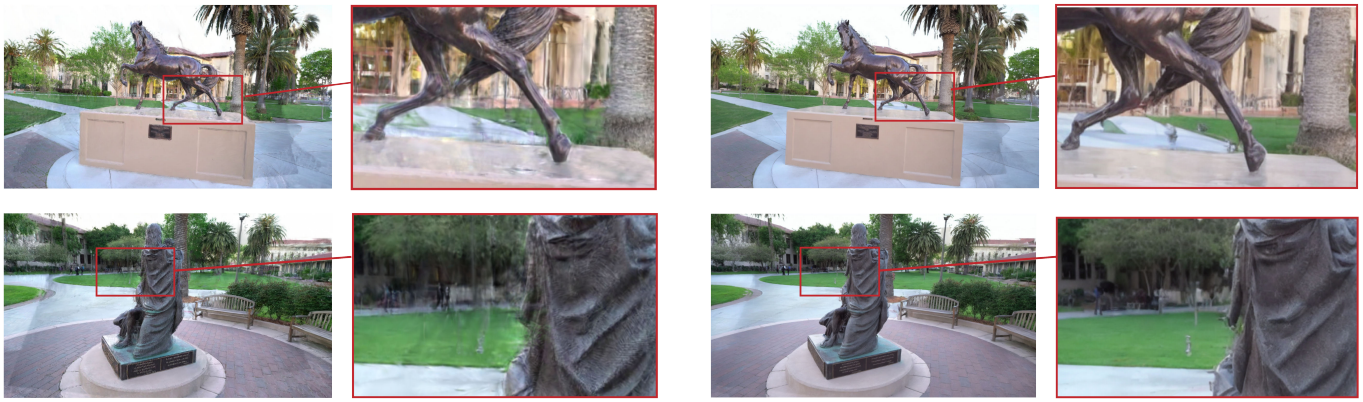
Effect of Multi-modal regularized Gaussian Reconstruction: We conducted ablation studies on the multi-modal regularization, incorporating depth, normal, and virtual-view regularization. The quantitative outcomes are detailed in Table VII. Following the integration of these regularization techniques, the novel view synthesis demonstrates improved results, indicated by higher PSNR and SSIM values, as well as finer details with lower LPIPS scores. With the implementation of multi-modal regularization, as depicted in Figure 13, the Gaussian-rendered images showcase smoother surfaces and reduced artifacts within the scene. Conversely, images lacking regularization exhibit black holes and sharp angles, diminishing the overall quality.

Effect of Iterative Gaussian Refinement: We further explore the usefulness of the iterative Gaussian refinement module. As illustrated in Figure 14, we present a comparison between

(a) Gaussian-rendered Image without multi-modal regularization

(b) Gaussian-rendered Image with multi-modal regularization

Fig. 13. **Qualitative Comparison between Gaussian-rendered images with and without multi-modal regularization.** Through multi-modal regularization, Gaussian-rendered images exhibit smoother surfaces and reduced artifacts within the scene. In contrast, images lacking regularization display black holes on houses, trees, and sharp angles on the ground that detract from the overall quality.



(a) Before Gaussian Repair

(b) After Gaussian Repair

Fig. 14. **Visual Comparison of Images before and after Gaussian Repair**: The images displayed on the left showcase Gaussian-rendered images derived from Coarse Gaussian Reconstruction Module. Noticeably, these images exhibit blurriness and artifacts. In contrast, the images on the right demonstrate a marked improvement after being repaired by our Gaussian Repair Model, showcasing a cleaner and higher-quality outcome.

the images before and after Gaussian Repair. The noticeable outcomes highlight that the repaired images exhibit enhanced details and a reduction in artifacts, emphasizing the effectiveness of the Gaussian Repair Model. Quantitative results before and after the Iterative Gaussian Refinement are also detailed in Table VII. While a marginal improvement in PSNR is noted, more intricate metrics such as LPIPS and SSIM demonstrate substantial enhancements. These findings align seamlessly with our primary objective of restoring intricate details within the images.

The number of input images: In Figure 15, we assess our method using different number of sparse input images N. We compare LM-Gaussian with the original 3DGS across view splits of growing sizes $K \in \{4, 8, 16, 24, 32\}$. Notably, in sparse view scenarios, our method consistently enhances the performance of 3DGS. Furthermore, as the number of views increases, we observe that our proposed techniques, leveraging extensive model priors for regularization and repair, diminish in significance. While LM-Gaussian's rate of quality enhancement slows with increasing views, 3DGS showcases a

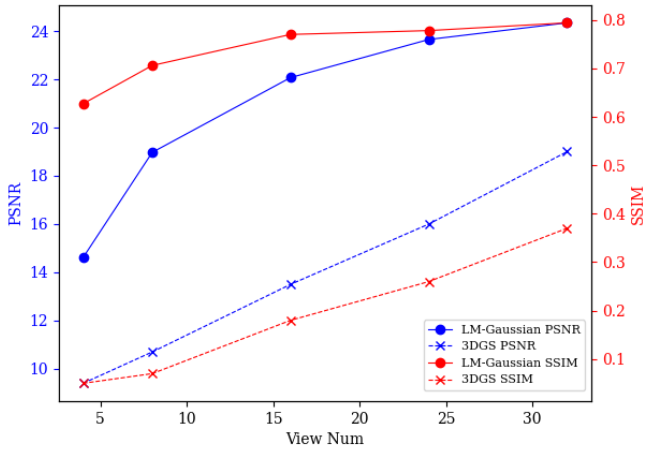


Fig. 15. Scalability of LM-Gaussian with input views. Our LM-Gaussian model demonstrates superior performance compared to the vanilla 3DGS, particularly evident in the notable improvements in PSNR and SSIM metrics.

steady linear progression.

VI. Conclusions

We introduce LM-Gaussian, a sparse-view 3D reconstruction method that harnesses priors from large vision models. Our method includes a robust initialization module that utilizes stereo priors to aid in recovering camera poses and reliable Gaussian spheres. Multi-modal regularizations leverage monocular estimation priors to prevent network overfitting. Additionally, we employ iterative diffusion refinement to incorporate extra image diffusion priors into Gaussian optimization, enhancing scene details. Furthermore, we utilize video diffusion priors to further improve the rendered images for realistic visual effects. Our approach significantly reduces the data acquisition requirements typically associated with traditional 3DGS methods and can achieve high-quality results even in 360-degree scenes. LM-Gaussian currently is built on standard 3DGS that only works well on static scenes, and we would like incorporate dynamic 3DGS techniques to enable dynamic modeling in the future.

References

[1] Rameen Abdal, Wang Yifan, Zifan Shi, Yinghao Xu, Ryan Po, Zhengfei Kuang, Qifeng Chen, Dit-Yan Yeung, and Gordon Wetzstein. Gaussian shell maps for efficient 3d human generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9441–9451, 2024.

[2] Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting. *arXiv preprint arXiv:2404.03613*, 2024.

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021.

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

[5] Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. Nope-nerf: Optimising neural radiance field with no pose prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4160–4169, 2023.

[6] 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. Generative novel view synthesis with 3d-aware diffusion models. In *ICCV*, 2023.

[7] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19457–19467, 2024.

[8] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14124–14133, 2021.

[9] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21476–21485, 2024.

[10] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*, 2024.

[11] Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3d using gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21401–21412, 2024.

[12] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. *arXiv preprint arXiv:2402.14650*, 2024.

[13] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation. In *European Conference on Computer Vision*, pages 264–280. Springer, 2022.

[14] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4, 2009.

[15] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022.

[16] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023.

[17] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5501–5510, 2022.

[18] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14346–14355, 2021.

[19] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[21] Liangxiao Hu, Hongwen Zhang, Yuxiang Zhang, Boyao Zhou, Boning Liu, Shengping Zhang, and Liqiang Nie. Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 634–644, 2024.

[22] Shoukang Hu, Tao Hu, and Ziwei Liu. Gauhuman: Articulated gaussian splatting from monocular human videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20418–20431, 2024.

[23] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[24] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular stereo and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21584–21593, 2024.

[25] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4220–4230, 2024.

[26] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, 2021.

[27] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5322–5332, 2024.

[28] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[29] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 21357–21366, 2024.

[30] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[31] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[32] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *2011 IEEE international conference on robotics and automation*, pages 3607–3613. IEEE, 2011.

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

[34] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20775–20785, 2024.

[35] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18537–18546, 2023.

[36] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8508–8520, 2024.

[37] Zhe Li, Zerong Zheng, Lizhen Wang, and Yebin Liu. Animatable gaussians: Learning pose-dependent gaussian maps for high-fidelity human avatar modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19711–19722, 2024.

[38] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6517–6526, 2024.

[39] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21644–21653, 2024.

[40] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5741–5751, 2021.

[41] Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, et al. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5166–5175, 2024.

[42] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21136–21145, 2024.

[43] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8576–8588, 2024.

[44] Xian Liu, Xiaohang Zhan, Jiaxiang Tang, Ying Shan, Gang Zeng, Dahua Lin, Xihui Liu, and Ziwei Liu. Humangaussian: Text-driven 3d human generation with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6646–6657, 2024.

[45] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024.

[46] Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3d geometry-aware deformable gaussian splatting for dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8900–8910, 2024.

[47] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

[48] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024.

[49] Andreas Meuleman, Yu-Lun Liu, Chen Gao, Jia-Bin Huang, Changil Kim, Min H Kim, and Johannes Kopf. Progressively optimized local radiance fields for robust view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16539–16548, 2023.

[50] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (ToG)*, 38(4):1–14, 2019.

[51] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[52] Arthur Moreau, Jifei Song, Helisa Dhamo, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. Human gaussian splatting: Real-time rendering of animatable avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 788–798, 2024.

[53] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3d scene representation via self-organizing gaussian grids. *arXiv preprint arXiv:2312.13299*, 2023.

[54] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

[55] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022.

[56] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20299–20309, 2024.

[57] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024.

[58] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Nerf for outdoor scene relighting. In *European Conference on Computer Vision*, pages 615–631. Springer, 2022.

[59] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[60] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 501–518. Springer, 2016.

[61] Ruizhi Shao, Jingxiang Sun, Cheng Peng, Zerong Zheng, Boyao Zhou, Hongwen Zhang, and Yebin Liu. Control4d: Efficient 4d portrait editing with text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4556–4567, 2024.

[62] Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. Splattingavatar: Realistic real-time human avatars with mesh-embedded gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1606–1616, 2024.

[63] Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. Language embedded 3d gaussians for open-vocabulary scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5333–5343, 2024.

[64] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[65] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. *Advances in neural information processing systems*, 34:12278–12291, 2021.

[66] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20675–20685, 2024.

[67] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

[68] Chen Wang, Xian Wu, Yuan-Chen Guo, Song-Hai Zhang, Yu-Wing Tai, and Shi-Min Hu. Nerf-sr: High quality neural radiance fields using supersampling. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 6445–6454, 2022.

[69] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9065–9076, 2023.

[70] Henan Wang, Hanxin Zhu, Tianyu He, Runsen Feng, Jiajun Deng, Jiang Bian, and Zhibo Chen. End-to-end rate-distortion optimized 3d gaussian representation. *arXiv preprint arXiv:2406.01597*, 2024.

[71] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20902–20911, 2024.

[72] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021.

[73] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024.

[74] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004.

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

[76] Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition*, pages 16210–16220, 2022.

[77] Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction. *arXiv preprint arXiv:2403.16292*, 2024.

[78] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024.

[79] Jamie Wynn and Daniyar Turmukhambetov. Diffusionerf: Regularizing neural radiance fields with denoising diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4180–4189, 2023.

[80] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024.

[81] Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. Sparsegs: Real-time 360 {\deg} sparse view synthesis using gaussian splatting. *arXiv preprint arXiv:2312.00206*, 2023.

[82] Dejia Xu, Ye Yuan, Morteza Mardani, Sifei Liu, Jiaming Song, Zhangyang Wang, and Arash Vahdat. Agg: Amortized generative 3d gaussians for single image to 3d. *arXiv preprint arXiv:2401.04099*, 2024.

[83] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, 2024.

[84] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20923–20931, 2024.

[85] Chen Yang, Sikuang Li, Jiemin Fang, Ruofan Liang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Gaussianobject: Just taking four images to get a high-quality 3d object with gaussian splatting. *arXiv preprint arXiv:2402.10259*, 2024.

[86] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8254–8263, 2023.

[87] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. *arXiv preprint arXiv:2312.00732*, 2023.

[88] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330. IEEE, 2021.

[89] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*, 2023.

[90] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4578–4587, 2021.

[91] Heng Yu, Joel Julin, Zoltán Á Milacski, Koichiro Niinuma, and László A Jeni. Cogs: Controllable gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21624–21633, 2024.

[92] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024.

[93] Yifei Zeng, Yanqin Jiang, Siyu Zhu, Yuanxun Lu, Youtian Lin, Hao Zhu, Weiming Hu, Xun Cao, and Yao Yao. Stag4d: Spatial-temporal anchored generative 4d gaussians. *arXiv preprint arXiv:2403.14939*, 2024.

[94] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024.

[95] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

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

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

[98] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (ToG)*, 40(6):1–18, 2021.

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

[100] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024.

[101] Shijie Zhou, Zhiwen Fan, Dejia Xu, Haoran Chang, Pradyumna Chari, Tejas Bharadwaj, Suya You, Zhangyang Wang, and Achuta Kadambi. Dreamscene360: Unconstrained text-to-3d scene generation with panoramic gaussian splatting. *arXiv preprint arXiv:2404.06903*, 2024.

[102] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21634–21643, 2024.

[103] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. In *CVPR*, 2023.

[104] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3d gaussian avatars. *arXiv preprint arXiv:2311.08581*, 2023.

[105] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting:

Fast and generalizable single-view 3d reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10324–10335, 2024.