GBR: Generative Bundle Refinement for High-fidelity Gaussian Splatting and Meshing

Jianing Zhang, Yuchao Zheng, Ziwei Li, Qionghai Dai, *Fellow, IEEE,* and Xiaoyun Yuan

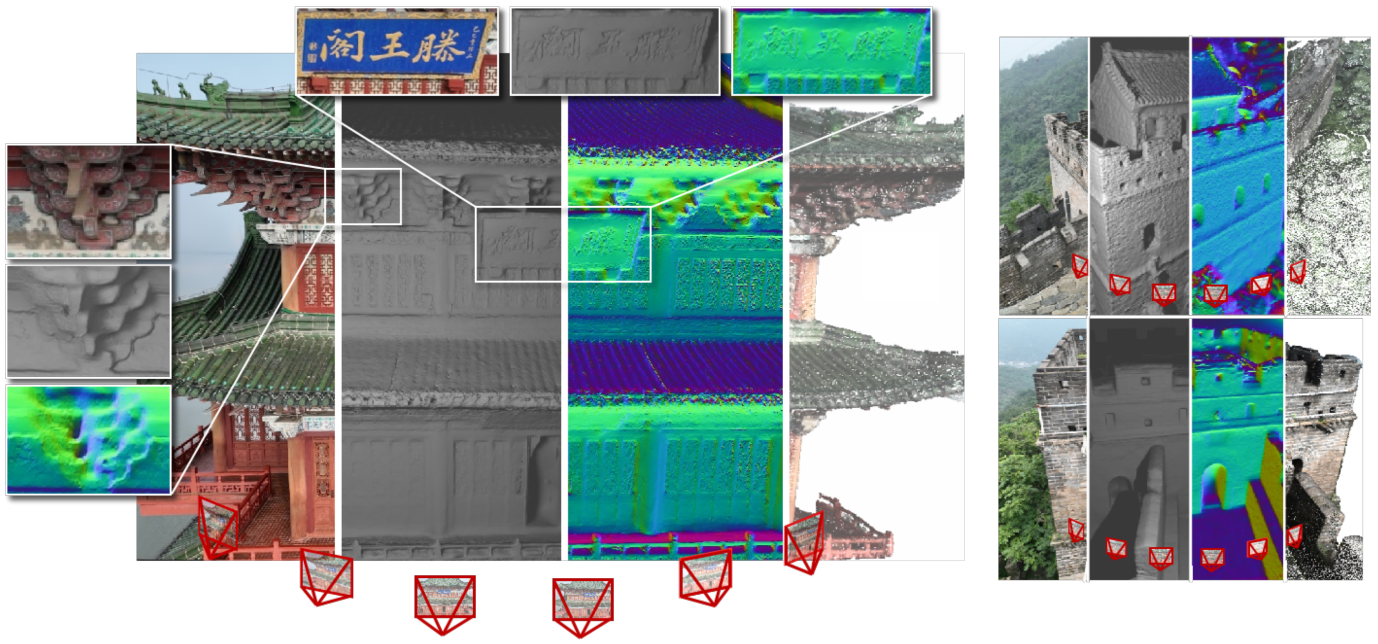

Fig. 1: Reconstruction results of GBR on large-scale real-world scenes: the Pavilion of Prince Teng and the Great Wall. The figure demonstrates our method's capability for accurate camera parameters estimation and detailed, high-fidelity surface reconstruction using **only 6 views**. From left to right, the images show novel view synthesis, mesh, normal map, and point cloud.

***Abstract*—Gaussian splatting has gained attention for its efficient representation and rendering of 3D scenes using continuous Gaussian primitives. However, it struggles with sparse-view inputs due to limited geometric and photometric information, causing ambiguities in depth, shape, and texture. we propose GBR: Generative Bundle Refinement, a method for high-fidelity Gaussian splatting and meshing using only 4–6 input views. GBR integrates a neural bundle adjustment module to enhance geometry accuracy and a generative depth refinement module to improve geometry fidelity. More specifically, the neural bundle adjustment module integrates a foundation network to produce initial 3D point maps and point matches from unposed images, followed by bundle adjustment optimization to improve multiview consistency and point cloud accuracy. The generative depth refinement module employs a diffusion-based strategy to enhance geometric details and fidelity while preserving the scale. Finally, for Gaussian splatting optimization, we propose a multimodal loss function incorporating depth and normal consistency, geometric regularization, and pseudo-view supervision, providing robust guidance under sparse-view conditions. Experiments on widely used datasets show that GBR significantly outperforms existing methods under sparse-view inputs. Additionally, GBR demonstrates the ability to reconstruct and render large-scale real-world scenes, such as the Pavilion of Prince Teng and the Great Wall, with remarkable details using only 6 views. More results can be found on our project page https://gbrnvs.github.io.**

***Index Terms*—Sparse-view Gaussian Splatting, Meshing, Generative Bundle Refinement, Neural Rendering.**

This work is supported in part by Natural Science Foundation of China (NSFC) under contracts No. 62271283 and No. 62231018, and Natural Science Foundation of Shanghai No. 22ZR1405500. (Corresponding author: Xiaoyun Yuan.)

J. Zhang is with the School of Information Science and Technology, Fudan University, Shanghai 200433, China (e-mail: 22110720080@m.fudan.edu.cn). Y. Zheng is with the School of Biomedical Engineering, Tsinghua University, Beijing 100084, China (e-mail: zhengyc23@mails.tsinghua.edu.cn). Z. Li is with the Key Laboratory for Information Science of Electromagnetic Waves (MoE), Department of Communication Science and Engineering, Fudan University, Shanghai 200433, China (e-mail: lizw@fudan.edu.cn). Q. Dai is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: qhdai@tsinghua.edu.cn). X. Yuan is with the MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yuanxiaoyun@sjtu.edu.cn).

I. Introduction

GAUSSIAN splatting has emerged as a promising method for achieving high-quality 3D reconstructions with lower computational and memory requirements compared to traditional voxel [1] or mesh-based approaches [2]. By optimizing the positions, rotations, scales, and colors of the 3D Gaussian primitives and combining alpha-blending, Gaussian splatting has achieved training time in minutes and real-time rendering. As a result, Gaussian splatting has enabled the generation of

3D content with broad applications in meta-universe[3], [4], autonomous driving [5], robotics manipulation [6], etc.

However, existing Gaussian splatting methods still require dense-view inputs (approximately 100 views) to achieve a good performance. When working with sparse-view inputs ($<$ 10 views), challenges such as background collapse and the presence of excessive floaters frequently arise, leading to inaccuracies in geometry reconstruction and degraded quality in novel view synthesis (NVS). For instance, the original Gaussian splatting pipeline[7] utilizes structure-from-motion (SfM) methods like COLMAP [8] to get the initial view poses and point cloud. Unfortunately, COLMAP struggles in sparse-view inputs, as it fails to generate a sufficiently dense and complete point cloud for the initialization of Gaussian primitives, thereby limiting the geometric accuracy and mesh fidelity. Recent methods [9], [10] also explored COLMAP-free initialization methods, but the dense-view inputs are still necessary.

To realize sparse-view, high-fidelity Gaussian splatting and meshing, we have to solve the following challenges: 1) **Geometry accuracy**. Conventional methods typically rely on Structure-from-Motion (SfM) to generate initial point clouds and view poses, achieving high accuracy but facing significant challenges with sparse-view inputs. Recent approaches [11], [12] address this issue by leveraging large vision models; however, the absence of explicit multi-view constraints introduces errors in geometry and view pose estimation. These inaccuracies hinder subsequent Gaussian splatting optimization, resulting in misaligned primitives and rendering artifacts. 2) **Mesh fidelity**. Additionally, high-accuracy point clouds alone are insufficient for reconstructing high-fidelity meshes. Geometric details are often lost during Gaussian primitives optimization due to the limited multi-view information available from sparse-view inputs. While some methods [13], [14] attempt to incorporate monocular depth information, inherent depth scale ambiguity compromises both geometry accuracy and fidelity. 3) **Insufficient Supervision**. Sparse-view inputs provide limited supervision for Gaussian primitives optimization, often causing the process to converge to local minima. To address this, it is essential to design effective loss functions and regularization terms that can better guide the optimization.

In this paper, we propose GBR: Generative Bundle Refinement, an effective framework to overcome the challenges mentioned above. Firstly, we propose neural bundle adjustment, which combines the widely used bundle adjustment optimizer with the neural network-based geometry estimator, such as DUSt3R[15]. The DUSt3R network can directly produce dense 3D point maps from 2D image pairs, and the conventional bundle adjustment optimizer enhances point map and view pose accuracy by incorporating explicit multi-view constraints. This combination effectively addresses the geometry accuracy challenge. Secondly, we introduce generative depth refinement, leveraging a diffusion model to incorporate high-resolution RGB information into the 3D point map. This process ensures scale-consistent integration, maintaining depth-scale accuracy while enhancing geometric detail and smoothness. Finally, we design a multimodal loss function incorporating confidence-aware depth loss, structure-aware normal loss, cross-view geometric consistency, and pseudo-view generation. Together, these components provide stronger supervision for Gaussian primitives optimization, leading to more accurate and robust reconstruction and rendering results. Our main contributions are as follows:

- We introduce a **comprehensive Gaussian splatting pipeline** that supports camera parameters recovery, depth/normal map estimation, novel view synthesis, and mesh reconstruction while only sparse-view inputs are required.
- We introduce **neural bundle adjustment**, which combines a deep neural network-based point map estimator with conventional bundle adjustment optimization to significantly enhance the accuracy of estimated camera parameters and the 3D point cloud.
- We propose **generative depth refinement** which uses a diffusion model to enrich geometric details while preserving depth scale, thereby boosting the fidelity of mesh reconstruction.
- We design a **multimodal loss function** that strengthens supervision, enabling more accurate and robust Gaussian primitives optimization.

II. Related Work

A. 3D Representations for Novel View Synthesis

Novel view synthesis (NVS) aims to create unseen perspectives of a scene based on a limited set of input images, a problem that has garnered substantial research interest [16]. NeRF [17] represents a prominent solution for photorealistic rendering by leveraging multilayer perceptrons (MLPs) to model 3D scenes, with spatial coordinates and view directions as inputs, and applying volume rendering techniques to synthesize images. While NeRF is renowned for producing high-quality renderings, it remains computationally intensive, requiring considerable training and inference time. Recent research has aimed to enhance either the quality [18], [19] or efficiency [20], [21] of NeRF; however, achieving an optimal balance between these aspects remains challenging.

As an alternative approach, 3D Gaussian splatting (3DGS) [7] addresses this dual objective by utilizing anisotropic 3D Gaussians [22] combined with differentiable splatting techniques, resulting in both efficient and high-quality reconstructions of complex scenes. However, 3DGS necessitates careful parameter tuning, particularly in adaptive density control, which is crucial for converting sparse SfM point clouds into dense 3D Gaussian distributions. Enhancing the quality and density of SfM results may further improve the effectiveness of 3DGS, potentially accelerating its optimization process.

B. 3D Representations for Surface Reconstruction

Building on the success of NVS in 3D representation, methods like NeRF and 3DGS have also been adapted for surface reconstruction tasks. NeRF-based approaches for surface modeling have incorporated occupancy fields [23] and signed distance fields [24], [25] to capture surface geometry. Although effective, the reliance of NeRF on MLP layers can be a bottleneck for both inference speed and flexibility in representation. To mitigate these limitations, recent approaches are moving away from MLP-heavy architectures, instead using discrete structures like points [26], [27] and voxels [28], [29], [30] to represent scene information more efficiently.

SuGaR [31] introduces a technique for extracting meshes from 3DGS by employing regularization terms to better align Gaussians with scene surfaces. This method samples 3D points from the Gaussian density field and applies Poisson surface reconstruction for meshing. While this improves geometric fidelity, the irregular shapes of 3D Gaussians can complicate smooth surface modeling, and the unordered structure of Gaussians makes the image reconstruction loss vulnerable to overfitting, potentially leading to incomplete geometry and surface misalignment issues.

To enhance multi-view consistency, methods such as 2DGS [32] compress the 3D volume into planar Gaussian disks, while GOF [33] uses a Gaussian opacity field to extract geometry through level sets directly. PGSR [34] transforms Gaussian shapes into planar forms, optimizing them for realistic surface representation and simplifying parameter calculations, such as normals and distance measures. Nonetheless, achieving consistent geometry across multiple views remains a challenge in 3DGS approaches.

C. Unconstrained 3D Representations

Both NeRF and 3DGS typically require extensive data from densely captured images and often depend on SfM preprocessing, such as with COLMAP [8], to estimate camera parameters and create initial sparse point clouds. This reliance on dense image captures and computationally heavy SfM limits their practical application, as errors in SfM can propagate through the 3D model, especially when input images lack sufficient overlap or exhibit low texture.

Several methods introduce regularization to support sparse-view scenarios. Techniques like RegNeRF [35], DGRNeRF [36], GeoRGS [37], and SparseNeRF [38] use depth priors to improve geometric accuracy, while DietNeRF [39] and ReconFusion [40] incorporate guidance from models such as CLIP [41] and diffusion networks [42]. Approaches like PixelNeRF [43] and FreeNeRF [44] utilize pre-training for few-shot NVS, while SparseGS [13], FSGS [14] and DNGaussian [45] employs monocular depth estimators for scenarios with limited views. However, these methods typically assume known camera poses derived from dense image captures.

To bypass the requirement for pre-determined camera poses, recent pose-free methods jointly optimize camera parameters with the 3D model using uncalibrated images. Strategies like NeRFmm [46] and BARF [47] adopt a coarse-to-fine approach for pose encoding. Innovations like Nope-NeRF [10], LuNeRF [48], and CF-3DGS [9] use depth priors to reduce the number of views required. However, they often assume densely captured video sequences, which can lead to optimization times similar to traditional SfM workflows.

Methods like InstantSplat [11] and LM-Gaussian [49] leverage DUSt3R [15] as an initializer, enabling estimation of camera parameters and a relatively dense point cloud. However, the noise inherent in DUSt3R's initialization significantly affects geometric precision, allowing only visually appealing novel view synthesis rather than accurate geometry reconstruction.

III. Preliminary on 3D Gaussian Splatting

The 3DGS algorithm reconstructs a scene by representing it as a collection of 3D Gaussian distributions, denoted $\{G_i\}$. Each Gaussian G_i is defined by the probability density function:

$$G_i(x|\mu_i, \Sigma_i) = \exp\left(-\frac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\right), \quad (1)$$

where $\mu_i \in \mathbb{R}^3$ represents the mean position of a point p_i in the scene, and $\Sigma_i \in \mathbb{R}^{3\times3}$ is the covariance matrix defining the spatial extent of the Gaussian. The covariance Σ_i can be factorized into a scaling matrix $S_i \in \mathbb{R}^{3\times3}$ and a rotation matrix $R_i \in \mathbb{R}^{3\times3}$ as follows:

$$\Sigma_i = R_i S_i S_i^\top R_i^\top. \quad (2)$$

This representation allows efficient rendering of the Gaussians via alpha blending. For each Gaussian, a transformation to the camera coordinate system and subsequent projection onto the 2D image plane are applied. With a transformation matrix W and the intrinsic camera matrix K, the parameters μ_i and Σ_i are transformed as:

$$\mu_i' = KW[\mu_i, 1]^\top, \quad \Sigma_i' = JW\Sigma_i W^\top J^\top, \quad (3)$$

where J is the Jacobian matrix approximating the affine projection transformation.

In order to create an image from a particular viewpoint, the color assigned to each pixel p is computed by blending the top K ordered Gaussians $\{G_i \mid i = 1, \cdots, K\}$ that intersect with p, according to the blending equation:

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

In this context, α_i is calculated by evaluating $G_i(u|\mu_i', \Sigma_i')$ multiplied by an opacity value learned for G_i. The variable c_i represents the color associated with G_i that can be learned. The Gaussians that intersect with p are arranged based on their depth from the current viewpoint. By leveraging differentiable rendering methods, it is possible to optimize all Gaussian parameters in an end-to-end manner.

The normal map from the current viewpoint is generated via alpha blending as:

$$\mathbf{N} = \sum_{i\in N} R_c^\top \mathbf{n}_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \quad (4)$$

where N is the number of all pixels and R_c is the rotation matrix from the camera frame to the global coordinate system and the Gaussian primitive normal vector $\mathbf{n}_i$ represents the direction of the minimum scale factor. The distance d_i of a point p_i from each Gaussian's plane to the camera is computed as:

$$d_i = \left(R_c^\top(\boldsymbol{\mu}_i - \mathbf{T}_c)\right) R_c^\top \mathbf{n}_i^\top, \quad (5)$$

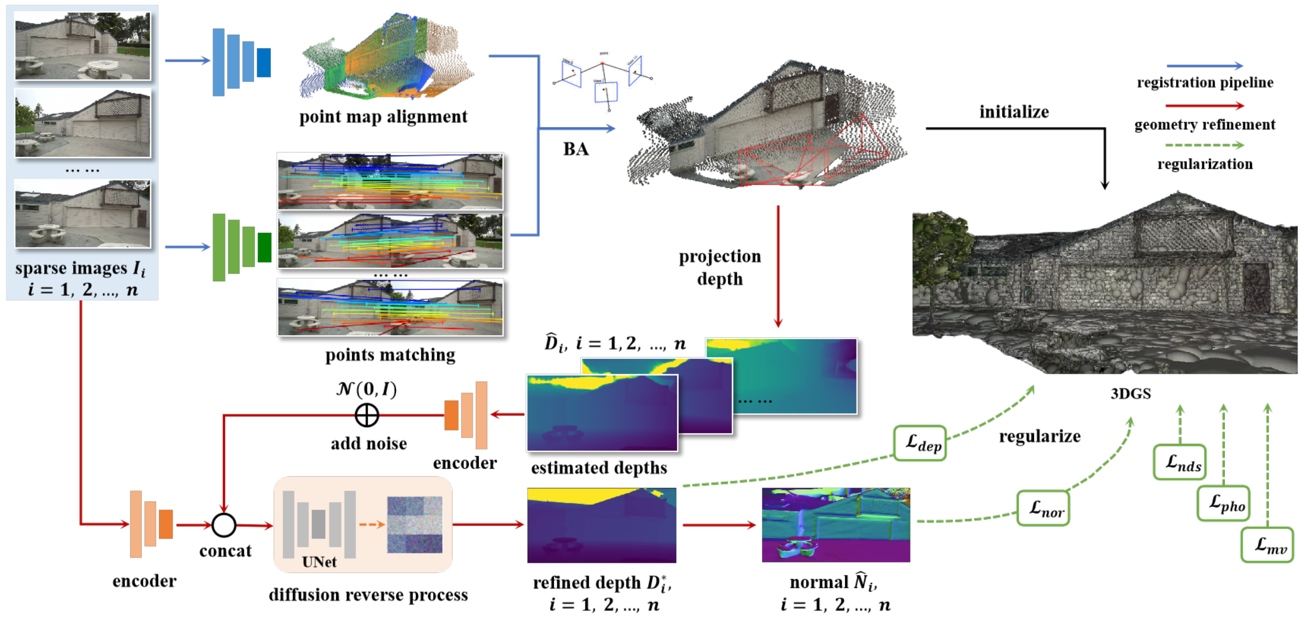

Fig. 2: Overview of our algorithm: Given unposed sparse-view inputs, we first employ neural Bundle Adjustment (neural-BA) to obtain a dense, accurate point cloud along with accurate camera parameters, providing a robust initialization for Gaussian splatting. Next, the dense point cloud is projected to obtain depth maps with accurate scale, and generative depth refinement is applied to enhance the depth details, resulting in scale-consistent, detail-rich depth and normal maps. Finally, a multimodal loss function incorporating depth, normal, pseudo-view synthesis, geometric consistency, and photometric loss is applied to optimize Gaussian splatting.

where $\mathbf{T}_c$ denotes the camera center in world coordinates, and $\boldsymbol{\mu}_i$ is the center of Gaussian G_i. Using alpha blending, the cumulative distance map from the current viewpoint is given by:

$$\mathcal{D} = \sum_{i \in N} d_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{6}$$

Following the depth calculation approach introduced in PGSR, once the distance and normal maps are rendered, the corresponding depth map can be derived by intersecting each pixel's viewing ray with the Gaussian plane. The depth calculation is given by:

$$\mathcal{D}(\mathbf{p}) = \frac{\mathcal{D}}{\mathbf{N}(\mathbf{p}) K^{-1} \tilde{\mathbf{p}}}, \tag{7}$$

where $\mathbf{p} = [u, v]^\top$ represents the 2D coordinates of a pixel on the image plane, $\tilde{\mathbf{p}}$ is the homogeneous representation of $\mathbf{p}$, and K denotes the camera's intrinsic matrix.

IV. Method

Gaussian splatting and meshing from sparse-view inputs encounter significant challenges, including geometry accuracy, mesh fidelity and limited supervision. To tackle these challenges, we introduce GBR: Generative Bundle Refinement, a framework designed to enhance Gaussian splatting and meshing through the following key components: 1) A neural bundle adjustment module, which combines the widely used bundle adjustment optimizer with the neural network-based geometry estimator, aiming to address the geometry accuracy challenge. 2) A generative depth refinement module that employs a diffusion model to integrate high-resolution RGB information into the point cloud, enhancing geometric details and smoothness. 3) A multimodal loss function, combining depth, normal, geometric consistency, synthesized pseudo-view supervision, and photometric losses, enhances reconstruction quality.

A. Neural Bundle Adjustment

Conventional multi-view reconstruction methods depend on bundle adjustment (BA)-based SfM and dense-input images to estimate camera poses and produce an initial point cloud. While for sparse-view inputs, SfM struggles to estimate accurate camera poses. The produced point cloud is also usually too sparse to support Gaussian splatting initialization and training. In recent years, neural network-based methods have provided a preliminary solution under sparse-view inputs. However, without explicit geometry constraints, these approaches cannot guarantee the accuracy of camera poses and point clouds. Hence, we propose the neural bundle adjustment that combines the widely used bundle adjustment optimizer with the neural network-based geometry estimator to improve the geometry accuracy and point cloud density under sparse-view inputs. It consists of the following 3 steps: (1) Neural dense initialization; (2) Bundle adjustment refinement with dual filtering; (3) Semantic-based local refinement.

1) Neural dense initialization: DUSt3R (Dense and Unconstrained Stereo 3D Reconstruction) leverages transformer-based architectures to unify stereo depth estimation into a

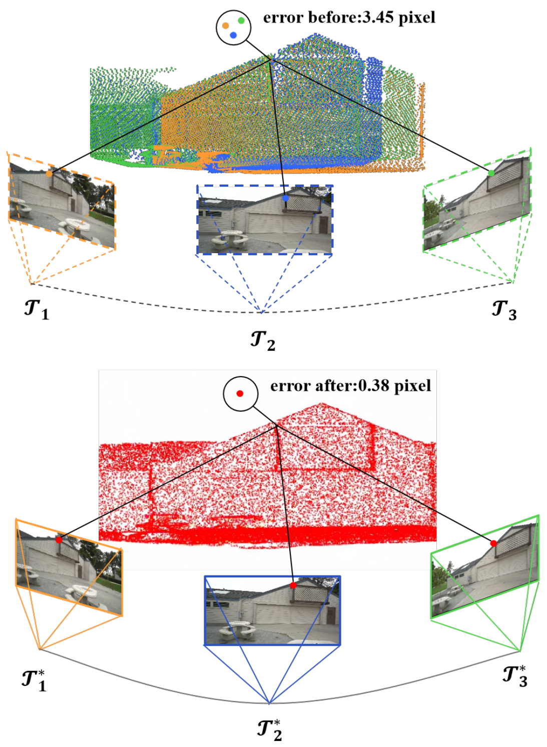

Fig. 3: Neural bundle adjustment illustration. After optimization, the point maps from different image views are well aligned.

single neural network. It enables direct 3D point cloud reconstruction from unposed image pairs $\{I_1, I_2\}$ as follows:

$$P^{W\times H\times 3}, C^{W\times H} = \text{DUSt3R}(I_1, I_2), \tag{8}$$

where W and H represent the width and height of the input images. The output includes $P \in \mathbb{R}^{W\times H\times 3}$, a 3D point map of I_1 with each pixel encoding its corresponding 3D position, and $C \in \mathbb{R}^{W\times H}$, a confidence map that quantifies the reliability of the reconstructed points.

The relative poses between images can be estimated through optimizing the following registration problem:

$$\arg\min_{T^{k,l},\sigma^{k,l}} \sum_{k\in K} \sum_{l\in K\setminus\{k\}} C^k \cdot C^l \|P^k - \sigma^{k,l} T^{k,l} P^l\| \tag{9}$$

where $\{P^k, P^l\}$ and $\{C^k, C^l\}$ denote the point maps and confidence maps of the image pair $\{I^k, I^l\}$. The purpose is to solve the transformation matrix $T^{k,l}$ and scaling factor $\sigma^{k,l}$ that aligns the point map from P^l to P^k. This process is applied across all image pairs $\{I^k, I^l\}$ to transform all the point maps to the same coordinate, generating a unified dense point cloud. Additionally, the focal length f of the image I (if not pre-calibrated) can be calculated by solving the following minimization problem:

$$\arg\min_f \sum_{i=0}^{W} \sum_{j=0}^{H} C_{ij} \left\| (i - \frac{W}{2}, j - \frac{H}{2}) - f \cdot \frac{(P_{i,j,x}, P_{i,j,y})}{P_{i,j,z}} \right\| \tag{10}$$

where C_{ij} is the confidence value at pixel position (i, j); and $P_{i,j,x}$, $P_{i,j,y}$, $P_{i,j,z}$ denote the x, y, and z components of the 3D point at pixel position (i, j).

Although the DUSt3R network generates an initial dense point cloud along with camera parameters, it does not explicitly account for epipolar geometric constraints. This oversight reduces the accuracy of the point cloud and the associated 3D-2D mappings, creating challenges for subsequent Gaussian splatting learning.

2) *Bundle adjustment refinement with dual filtering:* Therefore, we propose the bundle adjustment (BA) optimization with dual filtering to introduce the epipolar geometry constraints. Before BA, matching points can be obtained from the DUSt3R results:

$$\begin{aligned} \mathcal{M}_{k,l} &= \left\{(i,j) \;\middle|\; i = \text{NN}_k^{k,l}(j) \text{ and } j = \text{NN}_l^{l,k}(i)\right\}, \\ \text{NN}_k^{n,m}(i) &= \arg\min_{j\in\{0,\ldots,WH\}} \left\| P_j^{n,k} - P_i^{m,k} \right\|, \end{aligned} \tag{11}$$

where $P^{n,k}$ donates the point map P^n from camera n transformed into camera k's coordinate frame. $\mathcal{M}$ is matching points between images $\{I^k, I^l\}$, $NN_k^{n,m}$ denotes the nearest neighbor matching points extracted from the point maps. Through this, the nearest 3D point pairs can be found and used as corresponding 2D matching points. To improve matching quality, points are filtered based on the confidence map. By iterating over all pairs, all the 2D matching points $\mathcal{M}$ can be identified. Additionally, the inaccurate alignment in the initial point cloud also results in incorrect matches, causing failures in enforcing epipolar geometric constraints. Thus, we incorporate ROMA [50] to filter the matching points. That is to say, only matching points with high confidence values in DUSt3R results and ROMA results are preserved. The confidence thresholds of DUSt3R and ROMA are 3 and 0.05, respectively. This dual-filtering approach refines the location of 2D matching points and enables accurate bundle adjustment.

Finally, given the 2D matching points $\mathcal{M}$, intrinsic camera matrix $\mathcal{K}$, extrinsic camera matrix $\mathcal{T}$, and the initial point cloud X, the bundle adjustment is formulate as:

$$\begin{aligned} X^*, \mathcal{K}^*, \mathcal{T}^* &= \text{BA}(\mathcal{M}, X, \mathcal{K}, \mathcal{T}) = \arg\min\nolimits_{X,\mathcal{K},\mathcal{T}} \mathcal{L}_{\text{BA}}, \\ \mathcal{L}_{\text{BA}} &= \sum\nolimits_{i=1}^{N_I} \sum\nolimits_{j=1}^{N_x} v_i^j \left\| \mathcal{P}_i(\mathbf{x}^j, \mathcal{K}_i, \mathcal{T}_i) - y_i^j \right\|, \end{aligned} \tag{12}$$

where N_I is the number of input views, N_x is the number of matching points, v_i^j is a visibility term indicating whether point j is visible in view i, $\mathcal{P}_i(\mathbf{x}^j, \mathcal{K}_i, \mathcal{T}_i)$ denotes the projection of 3D point $\mathbf{x}^j$ onto the image plane of view i with camera parameter $\mathcal{K}_i$ and $\mathcal{T}_i$. y_i^j is the corresponding observed 2D location of point j in view i. Considering the computational complexity of global bundle adjustment, we typically limit the number of matching points selected from each view to no more than 50,000 for the calculations.

Upon optimizing the preserved point cloud and the camera parameters, we lower the confidence threshold for another round of bundle adjustment to optimize the position of the remaining points X' with the fixed camera parameters. We can get the final point cloud $\overline{X}^*$:

$$\overline{X}^* = X^* \cup \text{BA}(\mathcal{M}^*, X', \mathcal{K}^*, \mathcal{T}^*) \tag{13}$$

3) Semantic-based local refinement: The point cloud after the second round of bundle adjustment remains less dense than that initially provided by DUSt3R. Hence we first estimate a rigid transformation T to transform the initial point cloud to the optimized one:

$$T = \arg\min_{T} \|\overline{X}^{*} - TX\|. \tag{14}$$

Then, we refine the local regions with large misalignment before and after BA optimization. Local misaligned 3D regions are identified by measuring the point-wise distance $\overline{X}^{*} - TX$ between two point clouds. These points with large errors are then clustered using DBSCAN [51] and projected to 2D image views, serving as prompt points for the segmentation anything model (SAM) [52] in identifying local optimization areas. For these areas, we increase the number of matching points involved in BA optimization and perform additional BA optimization to achieve a dense and accurate point cloud.

B. Generative depth refinement module

Building upon the preceding methods, we have obtained a globally accurate and dense 3D point cloud. However, the resolution limitation of the DUSt3R network (maximum 512×512 pixels) restricts its ability to capture fine geometric details. Additionally, the limited overlap among sparse-view images leaves some 3D points insufficiently constrained, reducing quality in these regions.

Existing studies usually utilize the monocular depth estimation algorithms to assist sparse-view Gaussian splatting training. However, they suffer from scale ambiguity, providing only structural guidance, which is inadequate for detailed and accurate mesh reconstruction. Hence, we propose the generative depth refinement module, which integrates two key components: diffusion-based iterative depth refinement and scale-consistent depth integration.

1) Diffusion-based iterative depth refinement: Diffusion models achieve state-of-the-art results in image enhancement by learning to approximate the underlying data distribution through progressive denoising, which allows them to reconstruct images with high fidelity and fine-grained detail. Inspired by these approaches, we propose a diffusion-based algorithm to iteratively refine the depth details and accuracy. As demonstrated in Fig. 4, we modify Marigold's iterative process to leverage our geometry prior. The process combines a diffusion model and scale-consistent integration, consisting of the following key steps:

- Step 1: Initial depth map projection: We project the 3D point cloud into 2D depth maps. Limited by the processing resolution of the DUSt3R network, the initial depth map D^0 usually lacks fine details.
- Step 2: Latent encoding: The depth map and the corresponding RGB image are input to separate latent encoders, producing latent representations $z^{(D)}$ (depth) and $z^{(I)}$ (RGB).
- Step 3: Diffusion-based refinement: The latent depth representation $z^{(D)}$ undergoes an iterative refinement process guided by a diffusion model, operating entirely in the latent code space. Starting from the initial latent depth representation $z_0^{(D)}$, the process consists of two main stages: (1) In the forward process, Gaussian noise is gradually added to $z_0^{(D)}$ over T steps to generate a sequence of noisy latent representations $\{z_1^{(D)}, z_2^{(D)}, \ldots, z_T^{(D)}\}$. At each step t, this process is given by:
$$z_t^{(D)} = \sqrt{\bar{\alpha}_t} z_0^{(D)} + \sqrt{1-\bar{\alpha}_t}\epsilon,$$
where $\epsilon \sim \mathcal{N}(0, I)$, $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$, and $\{\beta_1, \ldots, \beta_T\}$ is the noise variance schedule. (2) In the reverse process, the noise is progressively removed from $z_T^{(D)}$ through T iterative steps. At each step t, the diffusion UNet refines the noisy latent depth $z_t^{(D)}$, concatenated with the RGB latent code $z^{(I)}$, to predict and subtract the noise. After T steps, this denoising process recovers the high-quality latent depth representation $z_0^{(D)}$.
- Step 4: Latent decoding: The refined depth latent is decoded into a high-resolution depth map, effectively addressing the resolution limitations of the DUSt3R network.

Our approach effectively ensures depth stability in the diffusion process while significantly improving efficiency. The original Marigold model requires 50 iterations, while our modified model only needs 10 iterations.

2) Scale-consistent depth integration: The diffusion process improves the depth details but may affect the depth-scale accuracy, which typically stems from two main sources: (1) The depth scale distribution gap between the training dataset of the diffusion model and our inputs. (2) Random noise introduced by the diffusion model.

To address the challenge, we add the scale-consist depth integration path to constrain the depth scale for each iteration (Fig. 4). More specifically, we design a spatially variant depth correction algorithm using a sliding window approach: for a pixel i within a local window ω_k, the corrected depth value is $D_i^* = a_k D_i + b_k$, where D is the depth map output by diffusion process, and D^* is the corrected depth map. D_i denotes the depth value of pixel i. a_i and b_i are the coefficients obtained by minimizing the following cost function:

$$\begin{aligned} E(a_i, b_i) &= \sum_{i \in \omega_k} \left((a_i D_i + b_i - D_i^0)^2 + \epsilon_i a_i^2 \right), \\ \epsilon_i &= \begin{cases} \epsilon_{\text{edge}}, & \text{if } |\nabla D_i^0| \geq \tau_e \\ \epsilon_{\text{smooth}}, & \text{if } |\nabla D_i^0| < \tau_e \end{cases}, \end{aligned} \tag{15}$$

Where D^0 is the initial depth map projected from the DUSt3R point cloud with accurate scale, ∇D_i^0 is its gradient map, and $\tau_e = 0.5$ is the gradient threshold. $\epsilon_i = 10^{-7}$ is a regularization weight to prevent a_i from becoming too large, helping to control the smoothness of the depth map. Sliding window mechanisms with windows size 25 and stride 1 and bilinear interpolation are adopted to compute the spatially variant correction parameters. The corrected output then serves as the input for subsequent rounds of diffusion. To reduce variability induced by diffusion noise, we generate multiple depth maps D' through repeated inferences. We then define a set $\mathcal{D}$ comprising depth maps that satisfy the alignment criteria:

$$\mathcal{D} = \{D' \mid ||D' - D^0||_2 \leq \tau_D\}, \tag{16}$$

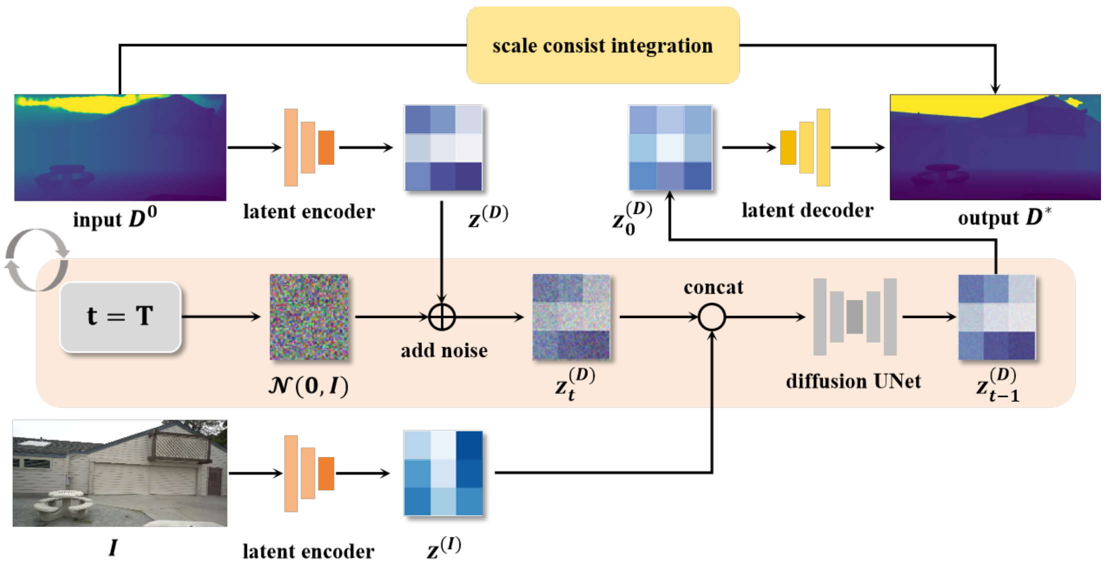

Fig. 4: Generative depth refinement module, including a diffusion model for iteratively refining the depth map and a scale-consist integration path to constrain the depth scale for each iteration.

where $\tau_D = 0.25$ is the alignment threshold. The final depth map D^* is computed by averaging the aligned depth maps in $\mathcal{D}$. The resulting depth maps, which are rich in detail and accurate in scale, provide high-quality guidance for the subsequent Gaussian splatting learning and surface reconstruction.

C. multi-modal supervision

In conventional 3D Gaussian splatting optimization, only photometric loss is considered, which usually fails for sparse-view inputs. Hence, we propose our multi-modal supervision composed of the 5 key components: confidence-aware depth supervision, structure-aware normal supervision, synthesized pseudo-view supervision, multi-view consistency geometric supervision, and photometric loss.

1) Confidence-aware depth supervision: During depth optimization, some areas exhibit significant depth variation before and after optimization, resulting in lower depth confidence. To address this, we implement a confidence-aware depth loss:

$$\mathcal{L}_{dep} = \frac{\sum_{x=1}^{W}\sum_{y=1}^{H} w_{\text{dep}}(x,y)\cdot|D^*(x,y)-\mathcal{D}(x,y)|}{\sum_{x=1}^{W}\sum_{y=1}^{H} w_{\text{dep}}(x,y)},$$
$$w_{dep}(x,y) = \frac{1}{1+\beta\cdot(|D^*(x,y)-D^0(x,y)|)/(|D^0(x,y)|)} \tag{17}$$

where $\beta = 0.1$ is a hyperparameter that controls the influence of depth variations and $\mathcal{D}$ is the depth map extracted from 3DGS. The sky region often introduces artifacts in mesh reconstruction, as the depth of the sky is almost infinite, making accurate depth estimation impractical. To solve this, we exclude sky regions from depth loss calculations.

2) Structure-aware normal supervision: An accurate normal map is vital for high-quality mesh reconstruction. For each pixel in the depth map, we select its four adjacent pixels—up, down, left, and right—and project them into the 3D space, resulting in a set of 3D points $\{\mathbf{p}_j \mid j = 0, \ldots, 3\}$. The normal vector is then defined as:

$$\hat{\mathbf{N}}(\mathbf{p}) = \frac{(\mathbf{p}_1-\mathbf{p}_0)\times(\mathbf{p}_3-\mathbf{p}_2)}{|(\mathbf{p}_1-\mathbf{p}_0)\times(\mathbf{p}_3-\mathbf{p}_2)|}. \tag{18}$$

Assume that the normal map derived from the depth map is $\hat{\mathbf{N}} \in \mathbb{R}^{H\times W\times 3}$, and $\hat{\mathbf{N}}(x,y)$ representing the normal vector at pixel (x,y). Then its local normal consistency $C(x,y)$ within a window is computed as:

$$C(x,y) = \frac{1}{w^2}\sum_{i=-\lfloor\frac{w}{2}\rfloor}^{\lfloor\frac{w}{2}\rfloor}\sum_{j=-\lfloor\frac{w}{2}\rfloor}^{\lfloor\frac{w}{2}\rfloor}\frac{\hat{\mathbf{N}}(x+i,y+j)\cdot\hat{\mathbf{N}}_m}{|\hat{\mathbf{N}}(x+i,y+j)||\hat{\mathbf{N}}_m|} \tag{19}$$

where $w = 3$ is the window size, $\hat{\mathbf{N}}_m$ represents the mean normal vectors within the window. A cosine similarity $C(x,y)$ function is then applied to convert the normal consistency from range [-1,1] to range [0,1]:

$$w_{nor}(x,y) = (1+C(x,y))/2 \tag{20}$$

and the structure-aware normal loss is finally defined as:

$$\mathcal{L}_{nor} = \frac{\sum_{x=1}^{W}\sum_{y=1}^{H} w_{nor}(x,y)\left|\mathbf{N}(x,y)-\hat{\mathbf{N}}(x,y)\right|}{\sum_{x=1}^{W}\sum_{y=1}^{H} w_{nor}(x,y)}, \tag{21}$$

where $\mathbf{N}$ denotes the normal map rendered from Gaussian splatting results.

3) Synthesized pseudo-view supervision: Since sparse-view inputs lack sufficient supervision from all view directions, the optimization of Gaussian primitives often converges to local minima. To address this, we generate n evenly distributed pseudo views between adjacent real views to provide additional supervision. In Gaussian primitives optimization, the weight of the virtual views is lower than that of real views.

More specifically, the pseudo-view depth maps are computed by averaging three sources:

$$D_t = (d_{n1} + d_{n2} + d_w)/3, \tag{22}$$

where d_{n1} and d_{n2} are the depth maps projected from the adjacent real depth maps, and d_w is the depth map projected from the point cloud. To ensure consistency, pixels from d_{n1} or d_{n2} that differ significantly (over threshold ϵ) from d_w are excluded from the averaging process.

The normal maps of the pseudo views can be obtained from Eq. 18. Directly projecting the point cloud to generate pseudo-RGB views produces unsatisfactory results due to the point cloud's relative sparsity compared to image pixel density. To address this, we employ Gaussian splatting to generate pseudo-RGB views. After 3,000 training iterations, the optimized Gaussian primitives can effectively render pseudo-view RGB images.

4) Geometric consistency loss: Geometric consistency can be categorized into single-view and multi-view consistency. Single-view consistency ensures that the rendered normals and depth maps are geometrically consistent within a single view. Multi-view consistency enforces that the rendered depth map adheres to epipolar geometry constraints across multiple views.

For single-view consistency, we define a normal-depth consistency loss inspired by 2DGS:

$$\mathcal{L}_{ndc} = \frac{1}{WH} \sum |\nabla D| \|\hat{\mathbf{N}} - \mathbf{N}\|_1, \tag{23}$$

where ∇D is the depth gradient normalized to the range [0, 1], $\hat{\mathbf{N}}$ is given by Eq. 18, and $\mathbf{N}$ denotes the normal map rendered from Gaussian splatting results. The depth gradient ∇D is used as a weighting factor in the loss function, emphasizing consistency in edge geometry.

For evaluating multi-view consistency, we propose a cycle-projection loss: pixels $\{x, y\}$ from an original view are first projected into 3D space using the rendered depth map, then reprojected into the 2D space of a neighboring view. A backward projection is then performed using the neighboring rendered depth map to project the pixels back to the original view, obtaining $\{x', y'\}$. By comparing the distances, we define the multi-view consistency loss:

$$\mathcal{L}_{mv} = \frac{1}{WH} \sum_{x}^{W} \sum_{y}^{H} \|x - x'\|^2 + \|y - y'\|^2 \tag{24}$$

The projection and back-projection matrices can be derived from the camera's intrinsic and extrinsic parameters of the two views.

For pseudo views, the neighboring view is defined as the closest real view, as the geometry of real views is typically more accurate. For real views, two options are considered: one involves randomly generating a nearby view to perform the forward and backward projections and calculate consistency, while the other involves using the nearest real view to compute geometric consistency. These two options are selected randomly.

5) Photometric loss: The photometric loss function incorporates both pixel-wise intensity differences and structural similarity:

$$\mathcal{L}_{pho} = (1 - \lambda)\mathcal{L}_1 + \lambda \mathcal{L}_{\text{SSIM}}, \tag{25}$$

$\lambda = 0.2$ is a hyperparameter that adjusts the relative importance of the two terms. $\mathcal{L}_1$ and $\mathcal{L}_{\text{SSIM}}$ are the L_1 loss and SSIM loss, respectively.

Finally, all losses are integrated as follows:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{nor} + \lambda_2 \mathcal{L}_{dep} + \lambda_3 \mathcal{L}_{ndc} + \lambda_4 \mathcal{L}_{mv} + \mathcal{L}_{pho}. \tag{26}$$

We set $\lambda_1 = 0.005$,$\lambda_2 = 0.005$,$\lambda_3 = 0.1$,and $\lambda_4 = 0.1$. Upon completing the training of the Gaussian primitives, we utilize the TSDF fusion method [53] to merge the depth maps from all views, resulting in the final mesh.

V. Experiments

A. Dataset and sparse view selection

Similar to previous algorithms, we selected the DTU [54], TNT [55], and MIP [18] datasets for testing our method. The DTU and TNT datasets are primarily used to evaluate geometric reconstruction performance, while the MIP dataset is employed to assess the algorithm's ability to generate novel views and to qualitatively showcase the mesh reconstruction results. For the DTU dataset, we selected 4 views for training and testing. This sparse-view configuration presents a significant challenge. For the larger scene datasets (TNT and MIP), we used 6 views as training data and an additional 6 neighboring views for testing. For the DTU and TNT datasets, we sampled the ground truth 3D geometry and only preserved those points visible from two or more views. For the TNT dataset testing, irrelevant point clouds, such as sky regions and background vegetation, were excluded to improve the quality of the results.

In addition, to further validate the effectiveness of our algorithm, we conducted experiments on the large-scale GigaNVS dataset [56], utilizing only six images per scene. The qualitative results on GigaNVS are shown in teaser and supplementary material.

B. Evaluation criteria

For the novel view synthesis task, we employed three widely used image evaluation metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). For surface quality evaluation, we assessed performance using the F_1 score (TNT) and Chamfer Distance (DTU). Additionally, we report the Absolute Trajectory Error (ATE) to evaluate the effectiveness of the camera parameters optimization. The real camera poses provided by the dataset are used as the ground truth.

C. Implementation details

In the context of BA, optimizing the camera's intrinsic and extrinsic parameters, as well as a subset of the point cloud, requires the selection of high-confidence point clouds

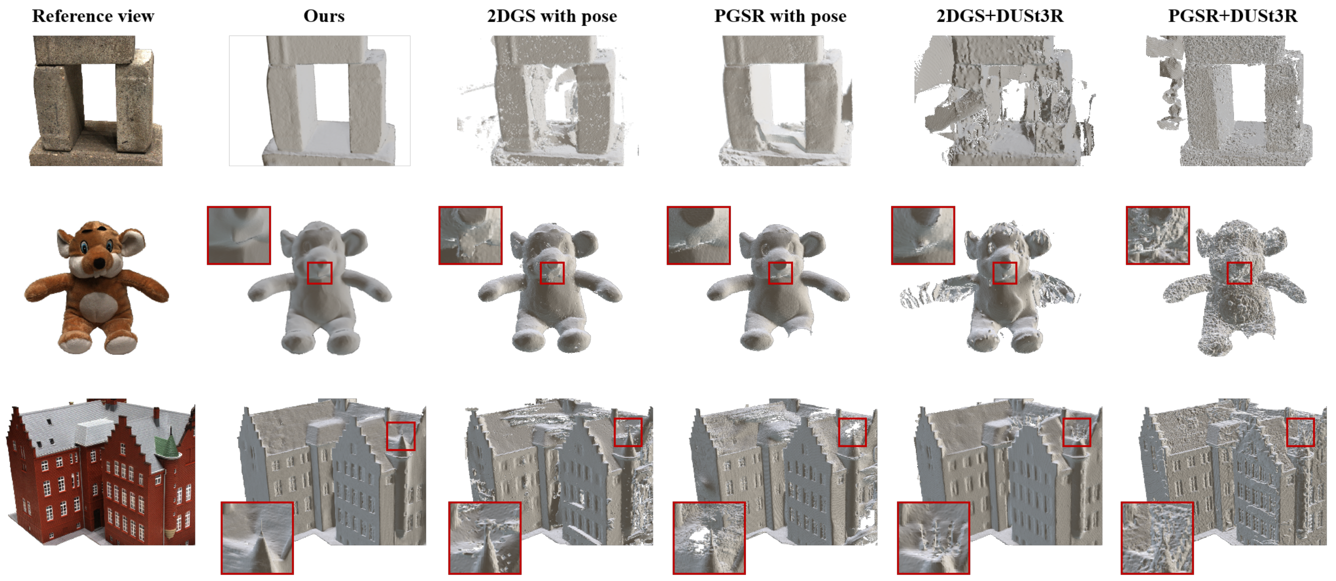

Fig. 5: Qualitative results using 4 input views on DTU dataset. Compared to DUSt3R initialization, our method achieves superior geometric accuracy. Compared to baseline methods using accurate poses computed by COLMAP, our approach generates more complete surface meshes.

for the calculations. Therefore, when the number of input views exceeds three, we set the confidence threshold to 3, meaning that a 3D point must be observed by at least three views. In other scenarios, the confidence threshold is set to 2. For point cloud filtering, we set the confidence threshold to 3.

For the training of Gaussian primitives, the basic training process and hyperparameters are similar to those of the original 3D Gaussian Splatting. However, a notable difference is that, due to the use of sufficiently dense point clouds for initialization, the number of training iterations can be significantly reduced. Specifically, we first train for 3,000 iterations, during which satisfactory results are often achieved. Subsequently, we render the RGB images for the pseudo views and perform depth optimization for these views before training for another 3,000 iterations. Comparatively, the conventional 3D Gaussian splatting typically requires around 300,000 iterations.

Generally, we adopt the same densification strategy as AbsGS [57]. However, due to the density of the initial point cloud, we do not apply the strategy during the first 1,500 iterations. The frequency of applying densification is also adjusted in subsequent iterations, occurring once every 500 iterations. All experiments were conducted using an NVIDIA RTX 4090 GPU.

D. Baselines

We combine DUSt3R [15] with other Gaussian splatting-based surface reconstruction algorithms as the primary baseline. For the optimization of DUSt3R, we employ the method of InstantSplat [11] to average the intrinsic parameters, which can improve the accuracy of the generated 3D point cloud. Additionally, we also conducted experiments using COLMAP combined with the Gaussian splatting-based surface reconstruction algorithms. However, this approach did not yield satisfactory results on the TNT dataset and is not included in our evaluations. For surface reconstruction algorithms, we select 2DGS [32], GOF [33], and PGSR [34]. We excluded SuGaR [31] from our evaluation due to its relatively high failure rate.

E. DTU dataset

Table I presents the quantitative results of the geometric evaluations for the DTU dataset. As discussed earlier, network-based geometry estimation methods, such as DUSt3R, suffer from low accuracy. While the conventional COLMAP method can produce highly accurate results, it is less robust for sparse-view inputs. For instance, COLMAP fails to reconstruct the 100 data with only 4-view inputs. While our method combines the advantages of these two methods, significantly outperforming the baseline methods in most cases. Only in a few specific scenarios, such as scenes 55, 93, and 114, where COLMAP can provide accurate extrinsic parameters, the baseline methods show marginal competitiveness with our method.

As qualitatively shown in Fig. 5, COLMAP struggles to produce a complete mesh, resulting in significant holes. DUSt3R, on the other hand, generates an inaccurate initial point cloud and camera parameters, which leads to a noisy and non-smooth mesh. In contrast, our method ensures both high accuracy and a dense initial point cloud. It remains robust even with extremely sparse viewpoints, yielding accurate, high-fidelity mesh reconstruction.

F. TNT dataset

Compared to the DTU dataset, the Tank and Temples (TNT) dataset collects outdoor real-world data, which presents more significant challenges. The numerical evaluation results in Table II demonstrate that our method holds a clear advantage

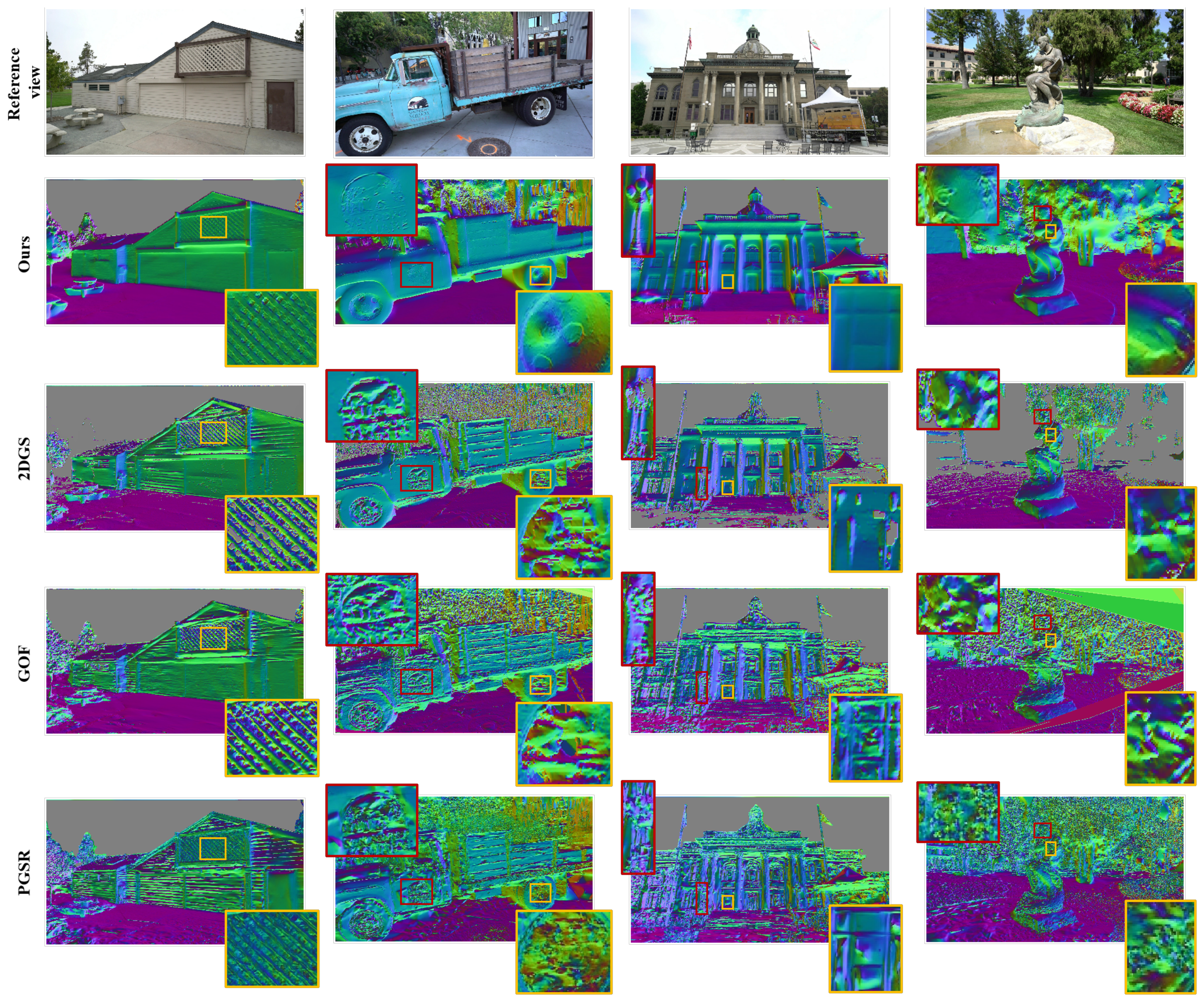

Fig. 6: Qualitative results using 6 input views on TNT dataset. The figure shows surface reconstruction quality using normal maps, with sky regions masked out. Our method generates smooth, detail-rich geometry, outperforming others that struggle with noise and fail to capture fine details. Additionally, our approach reconstructs both foreground and background geometry, unlike other methods restricted to foreground objects.

across all scenes. Fig. 6 shows the rendered normal maps. Our normal map is more complete and cleaner than the baseline methods, with better object details. For example, in our results, the eyes, nose, and fingers of the sculpture are successfully reconstructed, while the other methods fail to generate reasonable meshes and normal maps.

G. MIP dataset

We use the MIP dataset to test the performance of novel view synthesis. In addition to the baseline methods, we also evaluated two pose-free methods: InstantSplat and COLMAP-free 3D Gaussian splatting (CF-3DGS) [9]. InstantSplat uses DUSt3R for initialization but without extensive optimization, simply setting the focal length to an average value. CF-3DGS processes the input frames in a sequential manner and progressively grows the 3D Gaussians set by taking one input frame at a time.

As shown in Table III, our method outperforms the others in image rendering quality, and also estimates more accurate camera poses (ATE). Notably, PSNR improves by more than 2 dBs. The qualitative results, presented in Fig. 7, demonstrate that our method generates cleaner and smoother normal maps, leading to artifact-free, high-quality novel view synthesis results. Compared to the original DUSt3R (InstantSplat), the ATE error is reduced by 20%, significantly improving pose accuracy and enhancing the quality of surface reconstruction.

H. Ablations

Table IV presents the results of our ablation study, which evaluates the contribution of the neural-BA module, depth refinement module, depth loss, geometric consistency loss, and

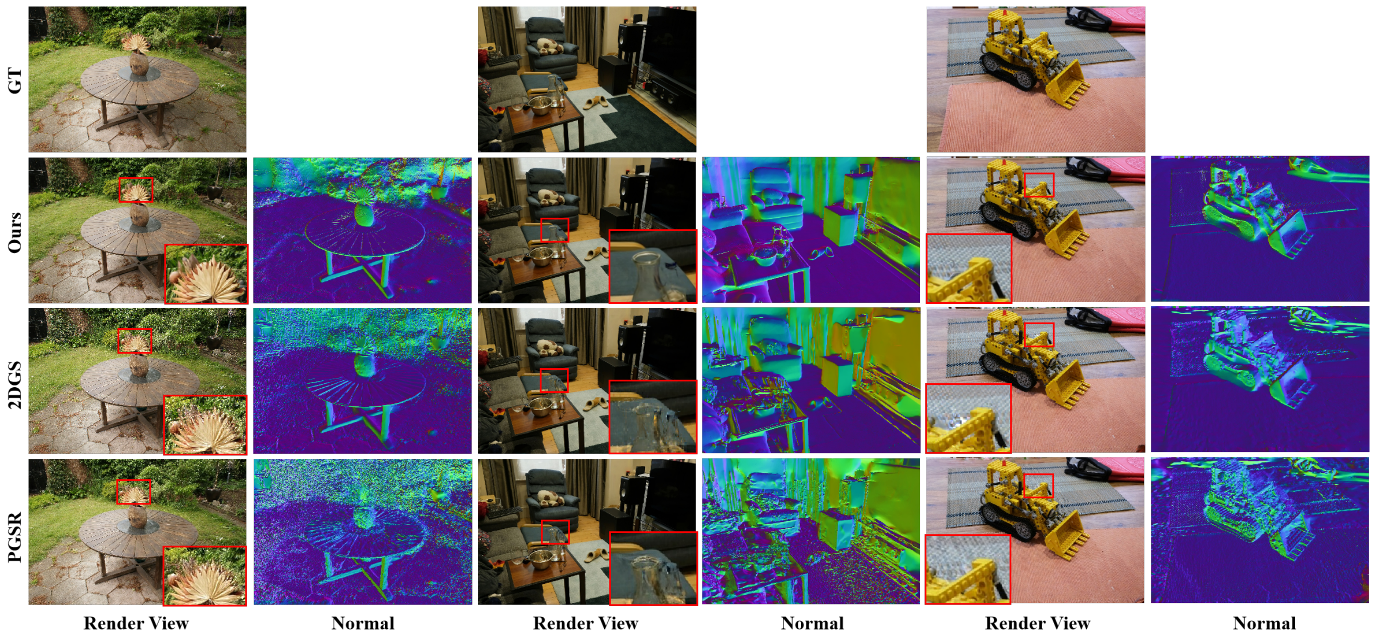

Fig. 7: Qualitative NVS results using 6 input views on the MIP dataset. The figure demonstrates the synthesized RGB images and normal maps of different methods. Our approach renders novel views with sharper details and fewer artifacts.

TABLE I: Quantitative comparison of methods on DTU data (Chamfer distance mm ↓). Red cells indicate the best performance, and light red cells indicate the second-best performance for each metric. '-' indicates experiment failure.

Method	**24**	**37**	**40**	**55**	**63**	**65**	**69**	**83**	**97**	**105**	**106**	**110**	**114**	**118**	**122**	**Mean**
2DGS+DUSt3R	0.846	4.127	4.197	4.594	5.133	5.457	5.030	1.020	1.453	4.985	4.530	0.727	4.560	5.084	1.670	3.561
PGSR+DUSt3R	0.948	4.740	3.836	4.371	1.623	2.002	4.628	0.641	1.349	4.696	0.669	0.726	4.334	4.379	6.451	3.026
GOF+DUSt3R	1.181	3.466	4.211	4.537	4.720	5.263	4.495	0.863	6.300	6.084	4.373	0.727	3.568	4.604	5.709	4.007
2DGS+COLMAP	1.327	0.877	1.239	0.565	1.008	0.953	1.365	1.020	1.453	1.053	1.268	-	0.676	1.406	1.670	1.134
PGSR+COLMAP	1.511	5.176	1.530	0.364	0.805	0.702	0.967	0.641	1.349	0.887	0.669	-	0.584	0.628	1.181	1.214
GOF+COLMAP	0.604	0.779	1.110	0.415	1.138	0.781	1.055	0.863	1.003	0.986	0.942	-	0.493	0.768	3.251	1.013
Ours	**0.413**	**0.642**	**0.555**	**0.281**	**0.505**	**0.553**	**0.525**	**0.611**	**0.667**	**0.587**	**0.406**	**0.664**	**0.393**	**0.535**	**0.561**	**0.526**

	2DGS	GOF	PGSR	Ours
Barn	0.1855	0.1517	0.1731	**0.2965**
Caterpillar	0.0808	0.0323	0.0755	**0.1998**
Courthouse	0.0446	0.0385	0.1112	**0.1452**
Ignatius	0.0636	0.0685	0.1315	**0.3126**
Meeting room	0.0314	-	0.0399	**0.1227**
Truck	0.1503	0.1183	0.1027	**0.2562**
Mean	0.0927	0.0682	0.1057	**0.2217**

TABLE II: Quantitative result of F_1 score↑ on Tanks and Temples dataset.

Method	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**ATE↓**
2DGS	19.831	0.482	0.358	0.222
GOF	20.967	0.530	0.310	0.222
PGSR	19.613	0.491	0.347	0.222
InstantSplat	18.755	0.510	0.420	0.126
CF-3DGS	15.516	0.2972	0.5508	0.275
Ours	**23.363**	**0.658**	**0.241**	**0.027**

TABLE III: Quantitative comparison of NVS task on MIP dataset. PSNR and SSIM are higher the better, while LPIPS is lower the better.

pseudo view loss. The Barn scene from the TNT dataset is used for testing. Each row corresponds to a different configuration of the system, with one key component excluded. Precision, recall, and F_1 score are used to show the impact of all the components. The results effectively highlight the critical role of all the components in the final performance.

VI. Limitations

Our method encounters challenges when dealing with scenes involving specular reflections and transparent objects. Addressing this limitation may require incorporating polarized imaging data or training advanced foundation models to better recognize such objects. Additionally, extending the method to support 4D reconstruction could be achieved in the future by integrating a motion-aware module.

	precision	recall	f-score
w/o Neural-BA	0.2287	0.2216	0.2251
w/o Depth Refinement	0.2836	0.2874	0.2855
w/o Depth Loss	0.2461	0.2520	0.2490
w/o Geometric consistency Loss	0.2786	0.2622	0.2701
w/o Pseudo views	0.2742	0.2791	0.2766
Full	**0.2975**	**0.2954**	**0.2965**

TABLE IV: Quantitative result of F_1 score↑ on Barn scene of TNT dataset.

VII. Conclusion

In this paper, we presented GBR: Generative Bundle Refinement, a framework designed for high-fidelity Gaussian splatting and meshing. By integrating neural bundle adjustment, generative depth refinement, and a multimodal loss function, GBR significantly improves geometric accuracy, detail preservation, and robustness in Gaussian splatting optimization. Our method effectively recovers camera parameters, generates dense point maps, and supports novel view synthesis and mesh reconstruction from sparse input images. We validated the performance of GBR on the DTU, TNT, and MIP datasets, demonstrating its superiority over state-of-the-art methods in both geometric accuracy and novel view rendering quality under sparse-view conditions. Finally, we demonstrated GBR's applicability to large-scale real-world scenes, such as the Pavilion of Prince Teng and the Great Wall.

References

[1] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *International journal of computer vision*, vol. 38, pp. 199–218, 2000.

[2] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, no. 4, 2006.

[3] N. Deng, Z. He, J. Ye, B. Duinkharjav, P. Chakravarthula, X. Yang, and Q. Sun, "Fov-nerf: Foveated neural radiance fields for virtual reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 11, pp. 3854–3864, 2022.

[4] W. Ye, H. Li, T. Zhang, X. Zhou, H. Bao, and G. Zhang, "Superplane: 3d plane detection and description from a single image," in *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2021, pp. 207–215.

[5] X. Zhou, Z. Lin, X. Shan, Y. Wang, D. Sun, and M.-H. Yang, "Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 634–21 643.

[6] Y. Zheng, X. Chen, Y. Zheng, S. Gu, R. Yang, B. Jin, P. Li, C. Zhong, Z. Wang, L. Liu *et al.*, "Gaussiangrasper: 3d language gaussian splatting for open-vocabulary robotic grasping," *arXiv preprint arXiv:2403.09637*, 2024.

[7] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics (TOG)*, vol. 42, pp. 1 – 14, 2023.

[8] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[9] Y. Fu, S. Liu, A. Kulkarni, J. Kautz, A. A. Efros, and X. Wang, "Colmap-free 3d gaussian splatting," *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20 796–20 805, 2023.

[10] W. Bian, Z. Wang, K. Li, J.-W. Bian, and V. A. Prisacariu, "Nope-nerf: Optimising neural radiance field with no pose prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4160–4169.

[11] Z. Fan, W. Cong, K. Wen, K. Wang, J. Zhang, X. Ding, D. Xu, B. Ivanovic, M. Pavone, G. Pavlakos, Z. Wang, and Y. Wang, "Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds," 2024.

[12] H. Yu, X. Long, and P. Tan, "Lm-gaussian: Boost sparse-view 3d gaussian splatting with large model priors," 2024. [Online]. Available: https://arxiv.org/abs/2409.03456

[13] H. Xiong, S. Muttukuru, R. Upadhyay, P. Chari, and A. Kadambi, "Sparsegs: Real-time 360 {\deg} sparse view synthesis using gaussian splatting," *arXiv preprint arXiv:2312.00206*, 2023.

[14] Z. Zhu, Z. Fan, Y. Jiang, and Z. Wang, "Fsgs: Real-time few-shot view synthesis using gaussian splatting," in *European Conference on Computer Vision*. Springer, 2025, pp. 145–163.

[15] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 697–20 709.

[16] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (ToG)*, vol. 38, no. 4, pp. 1–14, 2019.

[17] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[18] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5470–5479.

[19] ——, "Zip-nerf: Anti-aliased grid-based neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 697–19 705.

[20] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127

[21] J. Ding, Y. He, B. Yuan, Z. Yuan, P. Zhou, J. Yu, and X. Lou, "Ray reordering for hardware-accelerated neural volume rendering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 11, pp. 11 413–11 422, 2024.

[22] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "Ewa volume splatting," in *Proceedings Visualization, 2001. VIS'01.* IEEE, 2001, pp. 29–538.

[23] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3504–3515.

[24] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *arXiv preprint arXiv:2106.10689*, 2021.

[25] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4805–4815, 2021.

[26] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-nerf: Point-based neural radiance fields," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5438–5448.

[27] L. Lin, J. Zhu, and Y. Zhang, "Multiview textured mesh recovery by differentiable rendering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 4, pp. 1684–1696, 2023.

[28] H. Li, X. Yang, H. Zhai, Y. Liu, H. Bao, and G. Zhang, "Vox-surf: Voxel-based implicit surface representation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 3, pp. 1743–1755, 2022.

[29] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin, "Neuralangelo: High-fidelity neural surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8456–8465.

[30] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 651–15 663, 2020.

[31] A. Guédon and V. 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*, 2024, pp. 5354–5363.

[32] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH 2024 Conference Papers*, 2024, pp. 1–11.

[33] Z. Yu, T. Sattler, and A. Geiger, "Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes," *ACM Transactions on Graphics*, 2024.

[34] D. Chen, H. Li, W. Ye, Y. Wang, W. Xie, S. Zhai, N. Wang, H. Liu, H. Bao, and G. Zhang, "Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction," *arXiv preprint arXiv:2406.06521*, 2024.

[35] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. Sajjadi, A. Geiger, and N. 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*, 2022, pp. 5480–5490.

[36] S. Guo, Q. Wang, Y. Gao, R. Xie, L. Li, F. Zhu, and L. Song, "Depth-guided robust point cloud fusion nerf for sparse input views," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 9, pp. 8093–8106, 2024.

[37] Z. Liu, J. Su, G. Cai, Y. Chen, B. Zeng, and Z. Wang, "Georgs: Geometric regularization for real-time novel view synthesis from sparse inputs," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2024.

[38] G. Wang, Z. Chen, C. C. Loy, and Z. Liu, "Sparsenerf: Distilling depth ranking for few-shot novel view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 9065–9076.

[39] A. Jain, M. Tancik, and P. Abbeel, "Putting nerf on a diet: Semantically consistent few-shot view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5885–5894.

[40] R. Wu, B. Mildenhall, P. Henzler, K. Park, R. Gao, D. Watson, P. P. Srinivasan, D. Verbin, J. T. Barron, B. Poole *et al.*, "Reconfusion: 3d reconstruction with diffusion priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 551–21 561.

[41] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[42] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[43] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 4578–4587.

[44] J. Yang, M. Pavone, and Y. Wang, "Freenerf: Improving few-shot neural rendering with free frequency regularization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 8254–8263.

[45] J. Li, J. Zhang, X. Bai, J. Zheng, X. Ning, J. Zhou, and L. Gu, "Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization," *arXiv preprint arXiv:2403.06912*, 2024.

[46] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "Nerf–: Neural radiance fields without known camera parameters," *arXiv preprint arXiv:2102.07064*, 2021.

[47] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "Barf: Bundle-adjusting neural radiance fields," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 5741–5751.

[48] Z. Cheng, C. Esteves, V. Jampani, A. Kar, S. Maji, and A. Makadia, "Lu-nerf: Scene and pose estimation by synchronizing local unposed nerfs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 18 312–18 321.

[49] H. Yu, X. Long, and P. Tan, "Lm-gaussian: Boost sparse-view 3d gaussian splatting with large model priors," *arXiv preprint arXiv:2409.03456*, 2024.

[50] J. Edstedt, Q. Sun, G. Bökman, M. Wadenbäck, and M. Felsberg, "RoMa: Robust Dense Feature Matching," *IEEE Conference on Computer Vision and Pattern Recognition*, 2024.

[51] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[52] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[53] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[54] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, "Large-scale data for multiple-view stereopsis," *International Journal of Computer Vision*, pp. 1–16, 2016.

[55] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–13, 2017.

[56] G. Wang, J. Zhang, F. Wang, R. Huang, and L. Fang, "Xscale-nvs: Cross-scale novel view synthesis with hash featurized manifold," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 21 029–21 039.

[57] Z. Ye, W. Li, S. Liu, P. Qiao, and Y. Dou, "Absgs: Recovering fine details in 3d gaussian splatting," in *Proceedings of the 32nd ACM International Conference on Multimedia*, 2024, pp. 1053–1061.