IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

MVS-GS: High-Quality 3D Gaussian Splatting Mapping via Online Multi-View Stereo

BYEONGGWON LEE[1], JUNKYU PARK[1], KHANG TRUONG GIANG[2], SUNGHO JO[3*], and SOOHWAN SONG[1*]

[1]Division of AI Software Convergence, Dongguk University, Seoul 04620, Republic of Korea
[2]Team of 3D Vision, 42dot, Gyeonggi-do 13449, Republic of Korea
[3]School of Computing, KAIST, Daejeon 34141, Republic of Korea

Corresponding author: Soohwan Song (e-mail: songsh@dongguk.edu) and Sungho Jo (e-mail: shjo@kaist.ac.kr)

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2025-RS-2023-00254592) grant funded by the Korea government (MSIT).

ABSTRACT This study addresses the challenge of online 3D model generation for neural rendering using an RGB image stream. Previous research has tackled this issue by incorporating Neural Radiance Fields (NeRF) or 3D Gaussian Splatting (3DGS) as scene representations within dense SLAM methods. However, most studies focus primarily on estimating coarse 3D scenes rather than achieving detailed reconstructions. Moreover, depth estimation based solely on images is often ambiguous, resulting in low-quality 3D models that lead to inaccurate renderings. To overcome these limitations, we propose a novel framework for high-quality 3DGS modeling that leverages an online multi-view stereo (MVS) approach. Our method estimates MVS depth using sequential frames from a local time window and applies comprehensive depth refinement techniques to filter out outliers. The refinement method produces temporally consistent depths by checking sequential geometric consistency, enabling accurate initialization of Gaussians in 3DGS. Furthermore, we introduce a parallelized backend module that optimizes the 3DGS model efficiently, ensuring timely updates with each new keyframe. Experimental results demonstrate that our method outperforms state-of-the-art dense SLAM methods, achieving an average PSNR improvement of approximately 2 dB on indoor scenes. Moreover, our method reliably produces consistent 3D models in complex outdoor scenes, where existing methods often fail due to tracking errors and depth noise. It also reconstructs large-scale aerial scenes effectively, achieving an average PSNR gain of about 10.28 dB over existing methods.

INDEX TERMS online multi-view stereo, 3D gaussian splatting, neural rendering, dense SLAM, 3D modeling, depth estimation

I. INTRODUCTION

PRECISE 3D models are in high demand for various industrial applications, including digital twins and virtual or augmented reality. One of the most commonly used methods for 3D modeling is *multi-view stereo* (MVS) [1] [2] [3]. MVS generates high-quality 3D models by identifying dense correspondences among multiple images taken from different viewpoints. Recently, MVS has been combined with neural rendering techniques, such as neural radiance fields (NeRF) [4] and 3D Gaussian Splatting (3DGS), and extended to support novel view synthesis [5] [6]. NeRF represents a 3D scene using a multilayer perceptron (MLP) that predicts color and density along rays, enabling differentiable volumetric rendering optimized with multi-view supervision. To overcome NeRF's limitations in rendering speed and training time, 3DGS [7] [8] [9] has emerged as a promising alternative. 3DGS explicitly represents scenes using anisotropic Gaussians and achieves real-time, high-quality rendering through a differentiable tile-based rasterizer. However, conventional MVS remains an offline algorithm, requiring batch processing and substantial computational time. As a result, MVS has not been widely adopted in robotics or graphical applications that require real-time processing.

In this study, we address the problem of online 3D model generation for neural rendering using RGB-only frames. The problem refers to generating 3D models in real time from images, with the goal of enabling neural rendering without relying on depth sensors or precomputed geometry. Accurate 3D modeling with images is particularly difficult in online processing settings, as the absence of geometric information

FIGURE 1. **Rendering results of (a) Photo-SLAM [21] and (b) our method. The optimized Gaussian points and the estimated point cloud for each method are shown in the upper right and lower right, respectively.**

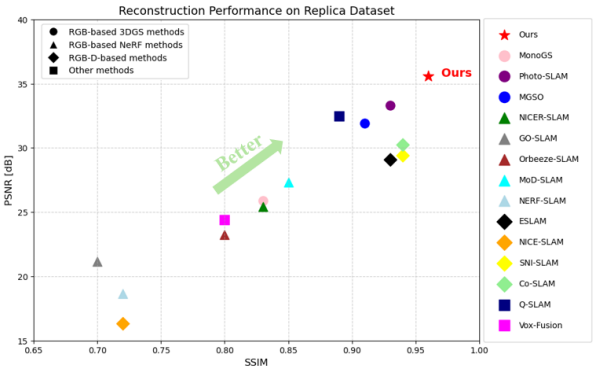


FIGURE 2. **Comparison between the proposed method (Ours) and state-of-the-art online mapping methods on the Replica dataset [26] evaluated with PSNR and SSIM.**

leads to scale ambiguity and uncertainty in depth estimation.

Dense simultaneous localization and mapping (SLAM) [10] [11] [12] tackles the challenge of online 3D modeling. Typically, dense SLAM methods estimate depth maps using motion stereo [10] [11] or depth prediction [13] and then integrate the depth maps online to construct a dense 3D model. Recently, many studies have adopted NeRF [14] [15] [16] [17] [18] [19] [20] or 3DGS [21] [22] [23] as map representations in dense SLAM, enabling real-time 3D modeling for rendering and view synthesis. However, existing methods primarily focus on estimating coarse 3D scenes rather than achieving detailed reconstruction. Most approaches rely on down-sampled images or lightweight networks for real-time computation, which significantly reduces the quality of the generated 3D models. Moreover, depth estimated solely from images is highly ambiguous due to factors such as motion blur, occlusion, or textureless regions. Despite this, current methods simply apply existing RGBD-based mapping techniques [24] [25] that rely on reliable depth, often resulting in noisy reconstructions. Therefore, a new approach is needed to estimate high-resolution, accurate depth maps and incorporate stronger geometric priors for high-quality mapping.

To address these challenges, we propose a novel framework that accurately estimates 3D information using an online MVS approach, which is then used to generate high-quality 3DGS models. The online MVS utilizes a sequence of frames within a local time window as source images and employs MVS networks [3] to estimate the depth map of the current frame. This method enables the real-time generation of precise, high-resolution depth maps. Additionally, since depth estimates derived solely from images can contain numerous outliers and inaccuracies, we implement a comprehensive depth refinement and filtering process. This step enhances the depth map by improving the consistency of sequentially estimated depth information, effectively removing outliers. As shown in Fig. 1, unlike existing methods [21] [22] [23] that rely on sparse or noisy depths to generate 3DGS models, our approach produces much denser and more accurate depth maps, facilitating the initialization of highly detailed Gaussian points.

Additionally, we developed an efficient mapping framework to enhance the learning efficiency of high-quality 3DGS within a limited timeframe. This framework includes an independent backend module focused on optimizing 3DGS models, which runs in parallel with the frontend module responsible for camera tracking and depth estimation. This parallel setup ensures that the 3DGS has sufficient time to refine its parameters before the next keyframe is processed. We also integrated *Generalized Exponential Splatting* (GES) [27] in place of the conventional 3DGS method [7], as GES requires fewer particles to represent sharp edges, leading to noticeable reductions in memory usage and training time.

The key contributions of this work are as follows:

- We propose a novel framework for high-quality 3DGS mapping, leveraging an online MVS method. The precise depth maps produced by MVS facilitate the initialization of accurate and dense Gaussian points.
- We introduce an online MVS method that incorporates a comprehensive depth filtering process. This approach sequentially estimates depths from incoming frames, integrates them to produce temporally consistent depths, and effectively filters out outliers.
- An efficient backend for 3DGS mapping is designed to operate in parallel with the frontend, ensuring sufficient time for model optimization. This backend also reduces the number of particles by initializing Gaussian points only for unexplored regions.
- The proposed framework has been evaluated on two benchmarks for indoor scenes [26] [28]. Unlike most previous studies that evaluate only on indoor scenes, we also tested the framework on challenging outdoor scenes [29] to demonstrate its generalization capability. Source code for our method is publicly available[1]

[1]https://github.com/lbg030/MVS-GS

II. RELATED WORKS

A. 3D MODELING VIA MULTI-VIEW STEREO

MVS reconstructs 3D models of scenes by identifying dense correspondences across multiple images [1]. Many studies have addressed challenging issues such as matching ambiguity and high computational complexity. Learning-based MVS methods [2] [3] have generally proven more effective than traditional approaches [1] in overcoming these challenges. Efforts have focused on improving performance by incorporating cascade cost volumes [2], or feature matching networks [30] [3].

NeRF [4] models a scene as a radiance field and optimizes it through differentiable volume rendering, resulting in photorealistic view synthesis. However, this process requires extensive per-scene optimization. To overcome this limitation, some studies [5] [6] have incorporated MVS techniques to generalize NeRF to unseen scenes. MVSNeRF [5] employs an explicit geometry-aware cost volume derived from MVS to model geometry in novel views. Liu et al. [6] introduced adaptive cost aggregation and a spatial-view aggregator to encode 3D context-aware descriptors for geometry-aware reconstruction.

Even though NeRF-based methods have achieved significant results, their performance is still limited by slow optimization and rendering speeds. As an alternative to NeRF, 3DGS [7] has recently been proposed and has gained considerable popularity in neural rendering. 3DGS explicitly represents scenes using anisotropic 3D Gaussians, allowing for real-time, high-quality rendering through a differentiable tile-based rasterizer. Similar to NeRF, several studies [8] [9] have attempted to apply MVS techniques to 3DGS to achieve generalized performance. GPS-Gaussian [8] combines iterative stereo-matching-based depth estimation with pixel-wise Gaussian parameter regression. MVSGaussian [9] also employs MVS to estimate depth and establishes a pixel-aligned Gaussian representation.

In [8] [9], the rich 3D information provided by MVS significantly enhances the rendering performance of the generated 3DGS. However, all of these methods [8], [9] are designed as offline processes, making real-time processing impossible. To overcome this limitation, developing an online 3DGS pipeline capable of real-time depth estimation and Gaussian parameter optimization is crucial. An online framework would enable efficient adaptation to dynamic scenes, expanding the practical applicability of 3DGS-based rendering techniques.

B. MONOCULAR DENSE SLAM

Traditional SLAM research [31] [32] has primarily focused on accurate camera localization, yielding impressive results. Recently, many studies have shifted their focus toward precise 3D mapping, leading to the increased popularity of dense SLAM. Classic methods used motion stereo with handcrafted features for dense [11] or semi-dense mapping [33]. DTAM [10] was among the first monocular dense mapping techniques, optimizing a photometric cost volume on the GPU. Similarly, REMODE [11] implemented probabilistic motion stereo using a recursive Bayesian estimation approach.

Many studies have integrated deep learning models into monocular dense SLAM. DeepFactors [34] compresses dense depth maps into a low-dimensional latent space using deep learning techniques, thereby reducing the computational burden of depth estimation. TANDEM [12] combines direct photometric visual odometry with MVSNet [35], subsequently fusing depth maps and extracting meshes from a truncated signed distance function. DROID-SLAM [36] employs an optical flow network to establish dense pixel correspondences and performs dense bundle adjustments. This approach achieves excellent trajectory estimation while simultaneously generating dense 3D models.

As NeRF [4] has advanced, many studies [14] [15] [16] [17] [18] have explored its potential for representing 3D spaces in rendering and novel view synthesis. Orbeez-SLAM [14] and iMODE [15] directly integrate the sparse features or semi-dense depths obtained from ORB-SLAM [31] into a NeRF model. Similarly, NeRF-SLAM [16] and GO-SLAM [17] utilize low-resolution depth maps estimated by DROID-SLAM for NeRF-based mapping. While some research [18] [19] has explored using depth prediction networks [13] for mapping, the accuracy of these predicted depths is often limited by the training dataset and does not match the precision of MVS-based depth estimates.

Recently, 3DGS has shown great promise in 3D modeling by addressing the limitations of NeRF and offering faster rendering speeds. Although only a few studies [21] [22] have applied 3DGS to dense SLAM, these have demonstrated notably superior performance compared to NeRF-based approaches. Photo-SLAM [21] generates Gaussian points from the sparse features extracted by ORB-SLAM and then densifies them using geometric information through a Gaussian-Pyramid-based learning approach. MonoGS [22] estimates camera poses directly from images rendered by 3DGS, bypassing explicit depth estimation and instead generating new Gaussians randomly from the rendered depths.

These methods [14] [15] [18] [21] [22] rely on sparse or inaccurate depth data to construct NeRF or 3DGS models, resulting in low-quality outcomes. Most approaches directly adopt RGBD-based mapping techniques [24] [25], which depend on reliable depth maps. However, the depths derived from RGB images are often imprecise and noisy, ultimately reducing performance. In contrast, our method utilizes MVS to estimate high-quality depths and applies stringent noise filtering techniques to generate Gaussian points. As a result, this approach produces a 3DGS model that delivers more accurate rendering compared to existing methods.

III. PROPOSED METHOD

Our goal is to reconstruct a 3DGS model online for high-quality rendering from an RGB image stream. Fig. 3 describes the overall architecture of the proposed modeling system. The system is mainly composed of two modules: the *frontend* and the *backend*, which operate in parallel as independent threads.

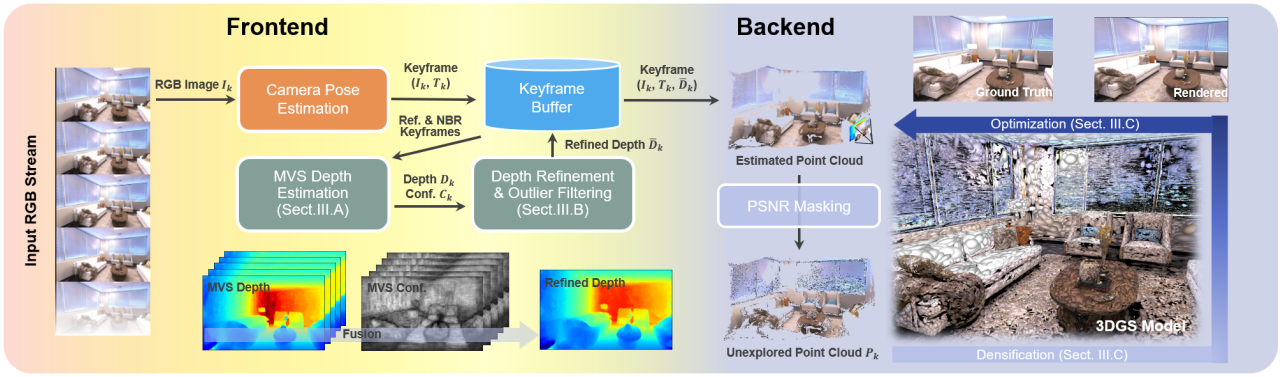


FIGURE 3. System overview: Our system consists of a frontend and a backend, both running in parallel. The frontend initially estimates the camera pose of keyframes using SLAM. It then estimates the depth map and confidence map of each keyframe based on MVS, refining the depth map by incorporating depth information from sequential keyframes. The backend generates new Gaussian points from the refined depth map and integrates them into the 3DGS model. The backend then continuously optimizes the 3DGS model.

The frontend first tracks camera poses $\{T_k\}$ of sequential input images $\{I_k\}$ by using the DROID-SLAM [36]. DROID-SLAM utilizes an optical flow network to predict dense pixel correspondences between adjacent frames. It then continuously performs dense bundle adjustment between keyframes to refine the camera poses. When a new keyframe is identified, the system performs MVS depth estimation by using the frame as a reference image (detailed in Section III.A). For each keyframe F_k it saves the image-pose pair (I_k, T_k) along with the estimated depth map D_k and confidence map C_k in a keyframe buffer. The estimated depth maps are relatively inaccurate and contain many outliers compared to offline MVS. Therefore, the system refines the depths and robustly filters out outliers based on previously estimated depth information (detailed in Section III.B).

The backend sequentially integrates the estimated depth maps into the 3DGS model (as detailed in Section III.C). It identifies unexplored regions on the depth map where 3DGS reconstruction is insufficient and converts only those depths into a point cloud P_k. This point cloud is then used to initialize new Gaussians $\{g_i\}$, which are subsequently added to the 3DGS model. The backend continuously optimizes the Gaussian parameters, running this process in parallel with the frontend while continuously appending new Gaussians from unexplored regions.

A. ONLINE MVS DEPTH ESTIMATION

We utilize the deep learning-based MVS method, MVSFormer [3], for estimating depth maps for each keyframe. MVSFormer uses a visual transformer to learn robust feature representations, achieving state-of-the-art results in MVS reconstruction. It features a cascade structure [2] that utilizes multiple smaller cost volumes instead of a single large one, which significantly reduces GPU memory usage and runtime. This cascading strategy enables the real-time generation of high-resolution depth maps, making it well-suited for our online modeling system.

Given a keyframe F_k, we use F_k as a reference frame and its neighboring keyframes $\{F_{k,n}^{nbr}\}$ as source frames to estimate the depth map D_k and confidence map C_k. $\{F_{k,n}^{nbr}\}$ are determined as the N_{nbr} consecutive keyframes before and after F_k. MVSFormer initially extracts multi-scale visual features from hierarchical vision transformers. These extracted features are used to construct cascade cost volumes at progressively finer scales using 3D CNNs [2]. The depths estimated by DROID-SLAM are used to determine the depth hypothesis range for the first-stage cost volume. It then incrementally refines the depth maps from coarse to fine by narrowing the depth range and reducing the number of hypotheses. Finally, MVSFormer predicts a depth map D_k and a confidence map C_k from a cost-volume by using the temperature-based method [3]. The method unifies the advantages of both regression-based depth and classification-based depth. The regression-based method (i.e., the expectation of depth probability) yields precise depth results. Conversely, using cross-entropy loss to optimize the network for depth classification offers a more reliable confidence estimation. By combining both methods, MVSFormer produces smooth and accurate depths and confidences.

Fig. 4 shows examples of reconstructed point clouds and estimated depth maps obtained from different depth estimation methods. Existing methods [16], [18], [19] typically construct 3D models using either low-resolution depth maps or predicted depth estimates. In Fig. 4a, a sparse point cloud is shown, resulting from the low-resolution depth maps generated via the dense optical flow of DROID-SLAM. Fig. 4b illustrates depth maps produced by the depth prediction method [13]; although the estimation appears accurate along object boundaries and planar surfaces, scale inconsistencies hinder proper alignment of the point cloud. In contrast, our method (Fig. 4c) utilizes an MVS-based approach, which estimates globally consistent depth and enables the reconstruction of a more complete and accurate 3D model.

B. DEPTH REFINEMENT AND OUTLIER FILTERING

The estimated depth maps often contain numerous outliers, especially in the background and boundary regions. Additionally, online depth estimation tends to produce more outliers compared to offline methods due to limited source views.

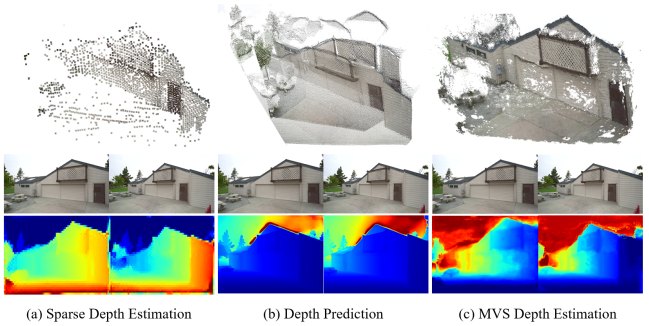


FIGURE 4. Examples of point clouds reconstructed from two estimated depth maps for each method. (a) low-resolution depth maps of DROID-SLAM, (b) learned depth prediction [13], (c) our online MVS depths.

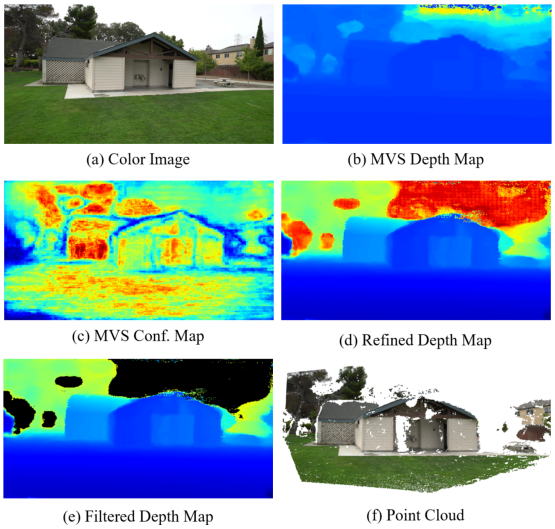


FIGURE 5. An illustration of the depth map refinement and filtering process: (b) the depth map and (c) the confidence map estimated by MVS are (d) refined using V-Fuse [37]. Our method then (e) filters outliers by checking the geometric consistency of the depths.

The source views can lead to incomplete coverage of the reference view and result in stereo matching errors from invisible pixels. To address these issues, our system incorporates additional steps to refine the depth maps and robustly filter outliers by leveraging sequentially estimated depth and confidence information.

Given the depth and confidence maps (D_k, C_k) of the reference keyframe F_k, as well as the set of depth and confidence maps $\{(D^{nbr}_{k,n}, C^{nbr}_{k,n})\}$ of the neighboring keyframes $\{F^{nbr}_{k,n}\}$, we integrate them to generate a refined depth and confidence maps $(\bar{D}_k, \bar{C}_k)$ for F_k. To solve this problem, we employ the depth map fusion network, *V-Fuse* [37]. It first calculates the mean and standard deviation of the input depth and confidence values to establish the depth hypothesis range for each pixel. It then constructs a visibility constraint volume by analyzing visibility factors such as support, occlusions, and free-space violations. This volume is regularized using a 3D convolutional network, after which the system generates the fused depth map through regression.

Next, we remove the outliers from the refined depth map. We first filter out low-confidence depths on $\bar{D}_k$ by applying a threshold to the confidence map $\bar{C}_k$. Then, it assesses the geometric consistency [38] of the depth $\bar{D}_k$ against the depths $\{D^{nbr}_{k,n}\}$ of $\{F^{nbr}_{k,n}\}$. For each neighboring keyframe $F^{nbr}_{k,n}$, the system wraps depths of $D^{nbr}_{k,n}$ onto the view of F_k. It calculates the relative depth differences between the wrapped depths of $F^{nbr}_{k,n}$ and the depths of F_k. If the relative depth difference is smaller than the defined threshold, the depth is considered geometrically consistent. Finally, depths that do not exhibit consistency across at least three views are filtered out. Fig. 5 illustrates the depth filtering process.

C. ONLINE 3DGS MAPPING

The backend module integrates the filtered depth maps $\bar{D}_k$ into a unified 3DGS model. To enhance efficiency, we employ the *generalized exponential splatting* (GES) [27] in place of the original approach [7] for constructing the 3DGS model. We have adapted the GES method to support online optimization of the 3DGS by sequentially inputting keyframes. The module continuously produces new Gaussian points from the incoming keyframes and updates their parameters through iterative processes. This operation is carried out in parallel with the frontend module.

Generalized Exponential Splatting. The 3DGS model represents the scene using a set of 3D Gaussians, $\mathcal{G} = \{g_i\}$, where each Gaussian point g_i is defined by a covariance matrix $\Sigma_i \in R^{3\times3}$, a mean $\mu_i \in R^3$, opacity $o_i \in [0,1]$, and color $c_i \in R^3$. Original 3DGS requires a large number of points to capture high-frequency details, making it inefficient for real-time mapping and rendering, particularly in large-scale scenes. To address this, we employ the more efficient method, GES, which is an extension of the generalized exponential function (GEF). Density function for each Gaussian with respect to a 3D point x is given by the GEF as:

$$g_i(x) = \exp\left\{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right\}^{\frac{\beta}{2}}$$

where β is a shape parameter that controls the splat's sharpness. GES treats this shape parameter as a trainable element across different frequency bands, leading to more adaptable and efficient rendering. This method effectively handles high-frequency components that posed challenges for the original 3DGS approach. GES also incorporates edge recognition via an edge-aware mask, which is generated using a difference of Gaussian filter.

To render an image $\hat{I}_k$ from an input pose $\hat{T}_k$, the 3D Gaussians are projected onto the image plane. The color $\hat{c}$ of each pixel p is determined by sorting the Gaussian points by depth and applying front-to-back alpha-blending, as follows:

$$\hat{c}(p) = \sum_{i\in N} c_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)$$

where α_i is calculated as the product of the opacity o_i and the GEF density g_i [27]. This method ensures that the closer

Gaussian points contribute more prominently to the final color, while the further points blend into the background.

Likewise, the rendered depth $\hat{d}$ can be obtained using the same projection method:

$$\hat{d}(p) = \sum_{i \in N} z_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)$$

where z_i represents the distance from the ray to the Gaussian mean.

The loss function used to train the GES is defined as:

$$\mathcal{L} = \lambda_{L1}\mathcal{L}_1 + \lambda_{ssim}\mathcal{L}_{ssim} + \lambda_{depth}\mathcal{L}_{depth} + \lambda_{\omega}\mathcal{L}_{\omega}$$

where $\mathcal{L}_1$ denotes the L1 loss computed between the rendered images and the original images. $\mathcal{L}_{depth}$ is also calculated between the rendered depth maps and the MVS depth maps, ensuring the preservation of the geometric structure. $\mathcal{L}_{ssim}$ represents the structural similarity loss [39], while $\mathcal{L}_{\omega}$ corresponds to the frequency-modulated loss derived from the edge-aware mask [27]. The $\mathcal{L}_{\omega}$ term ensures that the shape parameter is trained over a wide frequency range, allowing images to be rendered with fewer splats while enhancing edge sharpness.

Online Densification. The backend module incrementally densifies the 3DGS model by initializing new Gaussians at each incoming keyframe. We transform the filtered depth map $\bar{D}_k$ of a keyframe F_k into a 3D point cloud P_k, which is then used for densification. Since P_k contains a large number of 3D points, it would be inefficient to directly append all of them as new Gaussians. Therefore, we segment the depth map $\bar{D}_k$ into explored and unexplored regions, creating the point cloud P_k only from the depths in the unexplored regions. The explored region consists of areas where the current Gaussians already accurately represent the scene geometry, while the unexplored region includes areas where the Gaussians are either insufficient or newly observed. A pixel is classified as part of the unexplored region if its rendered color significantly differs from the original color. To identify such pixels, we compute the *peak signal-to-noise ratio* (PSNR) between the original and rendered images, marking those with a PSNR below a certain threshold as belonging to the unexplored region.

The extracted point cloud P_k might contain multiple points that are densely clustered in certain areas. To address this, we further reduce the number of points in P_k using a voxel grid filter. The filtered point cloud is then directly converted into new Gaussian points $\{g_i^{new}\}$, with each point assigned an identity covariance matrix Σ_i and $\beta = 2$. Finally, these new Gaussian points are added to the 3DGS $\mathcal{G}$.

Optimization. The backend module iteratively updates the parameters of the 3DGS model using differentiable rendering and gradient-based optimization [7]. It also performs adaptive density control to optimize the distribution of Gaussian points. This method dynamically adjusts the point density according to the level of detail required in different regions of the scene, ensuring efficient use of computational resources by assigning more Gaussians to high-frequency detail areas and fewer to low-frequency regions.

In each iteration, we use the current keyframe along with previously input keyframes for optimization. Due to limited computational time, instead of using all input keyframes, we select a subset of keyframes. Instead of selecting only recent keyframes, we randomly sample keyframes from the entire keyframe pool to create a keyframe subset. The module then performs an optimization on the selected subset to update the Gaussian parameters until the next new keyframe is received. This approach helps mitigate the forgetting problem, where the influence of earlier input frames gradually diminishes over time.

TABLE 1. Quantitative evaluation on the Replica RGB dataset

Method	Off0	Off1	Off2	Off3	Off4	Rm0	Rm1	Rm2	Avg.
PSNR ↑									
GO-SLAM	-	-	-	-	-	-	-	-	21.17
NICER-SLAM	28.54	25.86	21.95	26.13	25.47	25.33	23.92	26.12	25.41
GLORIE-SLAM	35.88	37.15	28.45	28.54	29.73	28.49	30.09	29.98	31.04
Q-SLAM	36.31	37.22	30.68	30.21	31.96	29.58	**32.74**	31.25	32.49
Photo-SLAM	36.99	37.52	31.79	31.62	**34.17**	29.77	31.30	33.18	33.29
MonoGS	32.00	31.21	23.26	25.77	23.85	23.53	25.00	22.42	25.88
MGSO	36.34	38.20	28.90	30.27	31.41	28.11	30.04	31.89	31.90
Ours	**41.22**	**42.24**	**34.12**	**34.67**	33.52	**32.18**	31.75	**35.89**	**35.70**
SSIM ↑									
GO-SLAM	-	-	-	-	-	-	-	-	0.70
NICER-SLAM	0.87	0.85	0.82	0.86	0.87	0.75	0.77	0.83	0.83
GLORIE-SLAM	0.97	**0.99**	**0.97**	**0.97**	**0.97**	**0.96**	**0.97**	**0.96**	**0.97**
Q-SLAM	0.94	0.94	0.90	0.88	0.89	0.83	0.91	0.87	0.89
Photo-SLAM	0.96	0.95	0.93	0.92	0.94	0.87	0.91	0.93	0.93
MonoGS	0.90	0.88	0.82	0.84	0.86	0.75	0.79	0.81	0.83
MGSO	0.95	0.96	0.90	0.91	0.93	0.82	0.87	0.92	0.91
Ours	**0.98**	0.98	0.95	0.96	0.95	0.95	0.93	**0.96**	0.96
LPIPS ↓									
GO-SLAM	-	-	-	-	-	-	-	-	0.41
NICER-SLAM	0.17	0.18	0.20	0.16	0.18	0.25	0.22	0.18	0.19
GLORIE-SLAM	0.09	0.08	0.15	0.11	0.15	0.13	0.13	0.14	0.12
Q-SLAM	0.13	0.15	0.20	0.19	0.18	0.18	0.16	0.15	0.17
Photo-SLAM	0.06	0.06	0.09	0.09	0.07	0.10	**0.08**	0.07	0.08
MonoGS	0.23	0.22	0.30	0.24	0.34	0.33	0.35	0.39	0.30
MGSO	0.24	0.25	0.31	0.27	0.25	0.35	0.29	0.26	0.28
Ours	**0.04**	**0.04**	**0.08**	**0.07**	**0.06**	**0.09**	0.11	**0.06**	**0.07**

TABLE 2. Quantitative evaluation on the TUM-RGBD dataset

Metrics	Method	f1/desk	f2/xyz	f3/off	Avg.
PSNR ↑	GO-SLAM	11.71	14.81	13.57	13.36
	MonoGS	19.67	16.17	20.63	18.82
	Photo-SLAM	20.97	21.07	19.59	20.54
	GLORIE-SLAM	20.26	**25.62**	21.21	22.36
	Ours	**24.25**	23.51	**24.81**	**24.19**
SSIM ↑	GO-SLAM	0.41	0.44	0.48	0.44
	MonoGS	0.73	0.72	0.77	0.74
	Photo-SLAM	0.74	0.73	0.69	0.72
	GLORIE-SLAM	**0.87**	**0.96**	0.84	**0.89**
	Ours	0.86	0.83	**0.86**	0.85
LPIPS ↓	GO-SLAM	0.61	0.57	0.64	0.61
	MonoGS	0.33	0.31	0.34	0.33
	Photo-SLAM	0.23	0.17	0.24	0.21
	GLORIE-SLAM	0.31	**0.09**	0.32	0.24
	Ours	**0.19**	0.17	**0.16**	**0.17**

IV. EXPERIMENTAL RESULTS

To verify the performance of our method, we conducted comparative experiments on online 3D modeling for neural rendering. We evaluated the modeling performance on both

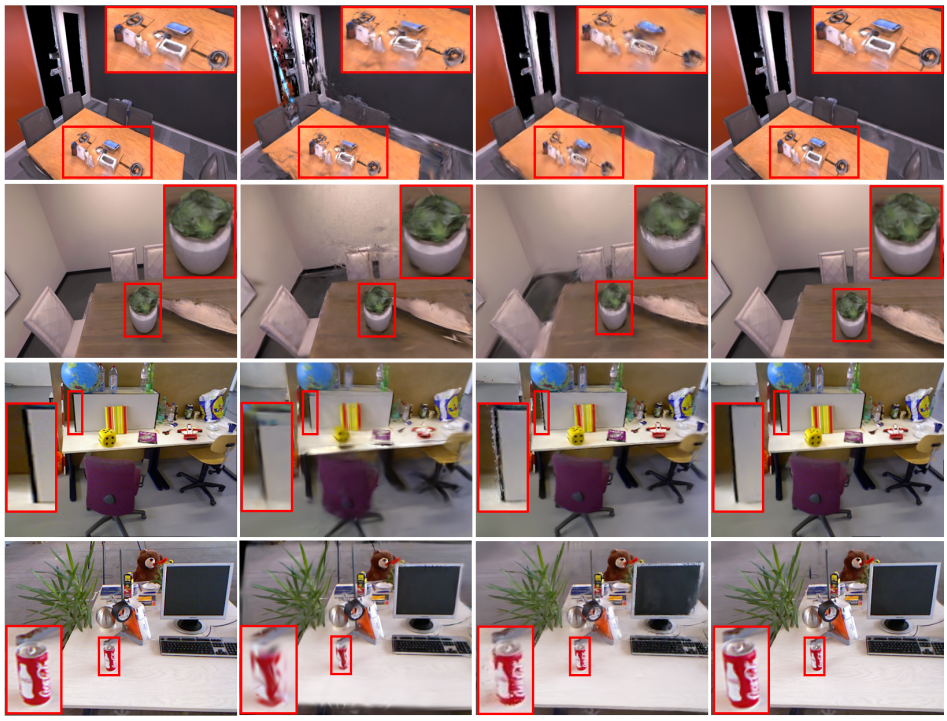
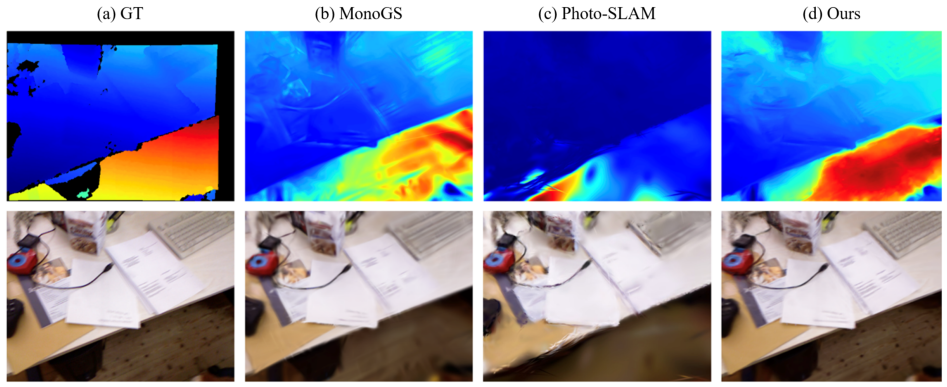


FIGURE 6. **Qualitative evaluation on the Replica RGB dataset (first and second rows) and the TUM-RGBD dataset (third and fourth rows).**



Depth L1 [cm] ↓	28.24	37.17	14.29
PSNR ↑	23.12	21.42	24.25

FIGURE 7. **Our system delivers superior scene reconstruction and rendering quality by utilizing enhanced depth estimation techniques and an optimized Gaussian learning approach. The results are averaged across all keyframes, with the experimental scene derived from the TUM-RGBD fr3 office dataset.**

indoor and outdoor scenes. Additionally, we performed an ablation study to assess the effectiveness of key components.

All experiments were performed on a desktop computer featuring an Intel i9-13900KS processor and an NVIDIA GeForce RTX 3090 Ti GPU. CUDA was employed for GPU acceleration in some processes, including time-sensitive rasterization and gradient computation, while the remaining components were implemented using PyTorch.

A. EVALUATION IN INDOOR SCENES

We evaluated the rendering performance of our method in indoor scenes using the Replica dataset [26] (8 sequences), the TUM RGBD dataset [28] (3 sequences), and the ScanNet dataset [40] (6 sequences). Although these datasets include RGBD frames, we focused solely on the RGB images. For MVS depth estimation, we used 384×512 images, while 3DGS was trained on the original resolution. We set the voxel resolution to $0.005m$, confidence threshold to 0.7, the number of neighboring keyframes N_{nbr} to 4, and PSNR threshold for masking to 40.

The performance of our method was compared against state-of-the-art monocular dense SLAM methods, including NeRF-based methods (GO-SLAM [17], NICER-SLAM [16],

TABLE 3. Rendering performance results on the ScanNet dataset.

Method	0000	0059	0106	0169	0181	0207	Avg.
				PSNR ↑			
GO-SLAM	15.74	13.15	14.58	14.49	15.72	15.37	14.84
MonoGS	16.91	19.15	18.57	20.21	19.51	18.37	18.79
GLORIE-SLAM	23.42	20.66	20.41	25.23	21.28	23.68	22.45
Ours	**23.91**	**22.60**	**21.65**	**25.56**	**22.18**	**24.22**	**23.35**
				SSIM ↑			
GO-SLAM	0.42	0.32	0.46	0.42	0.53	0.39	0.42
MonoGS	0.62	0.69	0.74	0.74	0.75	0.70	0.71
GLORIE-SLAM	**0.87**	0.87	0.83	0.84	**0.91**	0.76	**0.85**
Ours	0.82	**0.88**	**0.84**	**0.85**	0.80	**0.77**	0.83
				LPIPS ↓			
GO-SLAM	0.61	0.60	0.59	0.57	0.62	0.60	0.60
MonoGS	0.70	0.51	0.55	0.54	0.63	0.58	0.59
GLORIE-SLAM	0.26	0.31	0.31	**0.21**	0.44	0.29	0.30
Ours	**0.24**	**0.22**	**0.27**	**0.21**	**0.37**	**0.27**	**0.26**

TABLE 4. Rendering accuracy and efficiency comparison between NeRF-based methods (above the dashed line) and 3DGS-based methods (below the dashed line) on the Replica dataset.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	GPU Usage [GiB] ↓	FPS ↑
NICER-SLAM	25.41	0.83	0.19	21.30	0.13
GO-SLAM	21.17	0.70	0.41	18.50	9.21
GLORIE-SLAM	31.04	**0.97**	0.12	14.62	0.52
Ours	**35.70**	0.96	**0.07**	**11.27**	**11.56**

GLORIE-SLAM [19], and Q-SLAM [20]) and 3DGS-based methods (Photo-SLAM [21] and MonoGS [22]). For evaluating rendering performance, we utilized standard photometric metrics: PSNR, SSIM(Structural Similarity Index Measure) [39], and LPIPS (Learned Perceptual Image Patch Similarity) [41].

Tables 1 - 3 show the rendering accuracy on the Replica, TUM-RGBD, and ScanNet datasets, respectively. As illustrated in the tables, our method achieves the highest performance in average PSNR and LPIPS, while also securing either the top or second-best results in average SSIM. Notably, our method significantly outperforms MonoGS. MonoGS generates Gaussian points based on arbitrary depth estimates for unexplored scenes, which leads to inaccurate depth information and consequently produces low-quality 3DGS maps with poor rendering results. Additionally, our method surpasses Photo-SLAM in all metrics, demonstrating that generating Gaussian points from dense MVS depths is much more effective in producing high-quality 3DGS models compared to the sparse feature point-based initialization method employed by Photo-SLAM.

In addition to rendering accuracy, we evaluated the efficiency of each method by measuring GPU memory consumption and frame processing speed (FPS: the number of frames processed per second). Table 4 presents a comparison of both rendering quality and efficiency metrics for NeRF-based methods (NICER-SLAM, GO-SLAM), the 3DGS-based method (GLORIE-SLAM), and our proposed

(a) Photo-SLAM (b) Ours

FIGURE 8. Rendering results of (a) Photo-SLAM [21] and (b) our method on two aerial scenes. Each PSNR and SSIM represents the average result for the entire scene.

approach. As shown in Table 4, NeRF-based methods demonstrate significantly lower rendering performance and require substantially more GPU memory compared to 3DGS-based methods. For instance, NICER-SLAM achieves a PSNR more than 10 dB lower than our method while consuming nearly twice as much GPU memory. Furthermore, both NICER-SLAM and GLORIE-SLAM operate at less than 1 FPS, making real-time processing impractical. In contrast, our method achieves 11.56 FPS, offering significantly faster frame processing than all other methods. NeRF-based methods rely on computationally intensive volumetric rendering and implicit neural representations, which demand high GPU memory and slow down processing speed. Additionally, their reliance on low-resolution 3D volumes often leads to suboptimal reconstruction quality, resulting in lower PSNR.

Fig. 6 shows the qualitative comparison results of 3DGS-based methods. As shown in Fig. 6, our method produces high-quality rendering results with greater scene detail compared to the other methods. In particular, our method applies a comprehensive noise filter, resulting in significantly fewer artifacts.

Fig. 7 presents examples of depth maps and images rendered from 3DGS models generated by different methods, allowing for a comparative evaluation of their performance. As shown in the figure, our method not only produces cleaner rendered images compared to other approaches but also generates depth maps with significantly higher accuracy based on L1 loss. This demonstrates that the proposed method effectively estimates precise 3D structural information, which in turn contributes to improved image rendering performance.

B. EVALUATION IN OUTDOOR SCENES

This section assesses the generalization capability of our method by evaluating its performance on outdoor scenes, including aerial scenes [38] and the Tanks and Temples dataset [29]. Fig. 8 shows the rendering results along with the PSNR and SSIM performance of the generated 3DGS in two aerial scenes. The sequential images for these scenes were captured

(a) Family (PSNR: 25.98, SSIM: 0.91)

(b) Panther (PSNR: 30.99, SSIM: 0.95)

(c) Barn (PSNR: 27.41, SSIM: 0.89)

(d) Train (PSNR: 27.81, SSIM: 0.92)

(e) Horse (PSNR: 22.78, SSIM: 0.81)

FIGURE 9. **Rendering results of our method on the Tanks and Temples dataset. Our method produced exceptionally precise rendering results, closely mirroring real-world scenes. However, accuracy may drop in occluded or dynamic scenes, as shown in the red box area of (e).**

using a monocular camera mounted on a drone, which were then used for online 3DGS modeling. Both Photo-SLAM and our method successfully generated the 3DGS in the scenes, but MonoGS failed to produce the 3DGS model. As illustrated in Fig. 8, our method outperformed Photo-SLAM in terms of PSNR and SSIM, leading to more accurate rendering results. The disparity in performance between Photo-SLAM and our method is considerably more pronounced in outdoor scenes than in indoor scenes. Since MVS methods can accurately reconstruct large-scale scenes over wide ranges, our method proves to be significantly more effective in generating 3DGS models in aerial scenes.

Fig. 9 shows the rendering performance of our method in several outdoor scenes from the Tanks and Temples dataset. In these scenes, both Photo-SLAM and MonoGS were unable to track camera poses or generate 3DGS models due to insufficient continuous motion in the image frames. GLORIE-SLAM was able to track camera poses using DROID-SLAM; however, its depth prediction module failed to accurately estimate 3D information in untrained outdoor environments. As a result, it was ultimately unable to generate 3DGS models.

In contrast, our method was the only one capable of successfully tracking camera poses and accurately estimating depth maps in these scenes. Consequently, it was able to generate precise 3DGS models. As illustrated in Fig. 9, our approach delivered highly accurate rendering results that

TABLE 5. Ablation Study on the Replica "office 0"

Method	MVS Depth	Noise Filtering	Mask Update	PSNR	SSIM	LPIPS	FPS
Method-A	X	X	X	23.46	0.84	0.54	18.96
Method-B	O	X	X	40.74	0.97	0.05	13.72
Method-C	O	O	X	41.30	0.98	0.04	10.02
Method-D	O	O	O	41.22	0.98	0.04	11.56

closely resemble real images. These results demonstrate the effectiveness of the proposed method in leveraging online MVS for Gaussian point generation.

However, in particularly challenging scenes involving severe occlusions, motion blur, or dynamic objects, our method occasionally fails to produce fully accurate renderings. Although the Tanks and Temples dataset frequently includes heavy occlusion, our approach still demonstrated robust performance in most cases. Nevertheless, as shown in Fig. 9e, failure cases can occur in regions affected by dynamic objects or extreme occlusions. Addressing these limitations may require additional strategies, such as acquiring more views or filtering out dynamic objects during reconstruction.

C. ABLATION STUDY

We performed an ablation study on the Office0 scene from the Replica dataset to validate the effectiveness of the key components of our method. We measured the photometric metrics and FPS. Table 5 shows the performance variation of each key component (MVS depth estimation in Sect. III.A, depth filtering in Sect. III.B, and mask-based initialization of Gaussians in Sect. III.C) by comparing the four variants.

Method-A generates the 3DGS model directly from the low-resolution depth map of DROID-SLAM, resulting in high FPS but the lowest rendering performance. As shown in Fig. 4a, the low-resolution depth map produces a highly sparse Gaussian points, leading to a significant drop in performance. Method-C achieved the best rendering quality but had the lowest FPS. Compared to Method-B, the proposed noise filter significantly improved the performance of Method-C. Fig. 10 shows the qualitative comparison results with (Method-C) and without (Method-B) applying the depth refinement filter. As illustrated in the figure, applying the refinement filter results in significantly less noise and enables more detailed and cleaner reconstruction of objects. Interestingly, while Method-D slightly reduced rendering quality compared to Method-C, it achieved a higher FPS. This suggests that initializing Gaussian points only in unexplored regions is an effective strategy.

To evaluate the real-time performance of the proposed system, we measured the execution time of each module. Table 6 presents the average runtime and invocation interval for each component. All modules operate within their respective invocation intervals, indicating the absence of performance bottlenecks. In particular, the 3DGS optimization module runs asynchronously from the frontend processes (e.g., tracking and MVS), maintaining consistent mapping quality without interrupting the overall pipeline. This architecture ensures both computational efficiency and robustness, enabling stable operation in complex environments.

TABLE 6. The average runtime and invocation interval of each module of our method on the Replica "office 0". Modules that share the same invocation interval are considered interdependent, while the others operate independently.

Module	Tracking	MVS	Depth Refinement	3DGS Mapping	3DGS Opt.
Runtime (sec)	0.048	0.206	0.182	0.013	0.181 (per frame)
Invocation Interval (sec)	0.050	1.010		Consistent Optimization	

V. LIMITATIONS AND DISCUSSION

The proposed method achieved outstanding performance on various indoor benchmark datasets. Notably, by leveraging MVS techniques, our method effectively extracts highly accurate 3D information, even from large-scale environments. This capability enabled our method to successfully perform accurate 3D modeling in challenging outdoor scenes, where other existing approaches typically fail. As illustrated in Fig. 8, prior methods such as MonoGS and Photo-SLAM struggle to form adequate Gaussian representations in wide-field outdoor environments. In contrast, our approach consistently produces complete and robust 3DGS models through precise depth estimation and continuous optimization of Gaussians. Furthermore, experiments conducted on the Tanks and Temples dataset (Fig. 9) show that existing methods frequently encounter camera tracking failures or inaccurate initialization of Gaussians. Our method, however, maintains robust reconstruction results, empirically demonstrating its technical superiority. Despite these achievements, our proposed method still has several significant limitations.

First, our method focuses on precise reconstruction rather than rapid depth estimation; therefore, we adopted MVSFormer [3], which provides state-of-the-art performance. In our setup, MVSFormer can estimate depth maps for approximately five keyframes per second, which is sufficient for online processing in our system. Since Droid-SLAM extracts a keyframe every 1.3 seconds on average in the Replica dataset, depth estimation does not pose a bottleneck for our approach. Several studies have introduced efficient MVS network models [42], [43] that bypass the use of 3D convolutional neural networks for cost-volume regularization. Since these models can calculate depth maps much faster than MVSFormer, they could serve as alternatives if faster computation time is required.

Second, our method relies solely on keyframe information to train the 3DGS model. However, because keyframes do not fully capture the entire scene, this can result in incomplete reconstructions. Improving the generalization capability of the 3DGS model requires training with frames taken from diverse viewpoints. A promising direction for future research would be to identify non-keyframes that enhance modeling performance and integrate them into the training process.

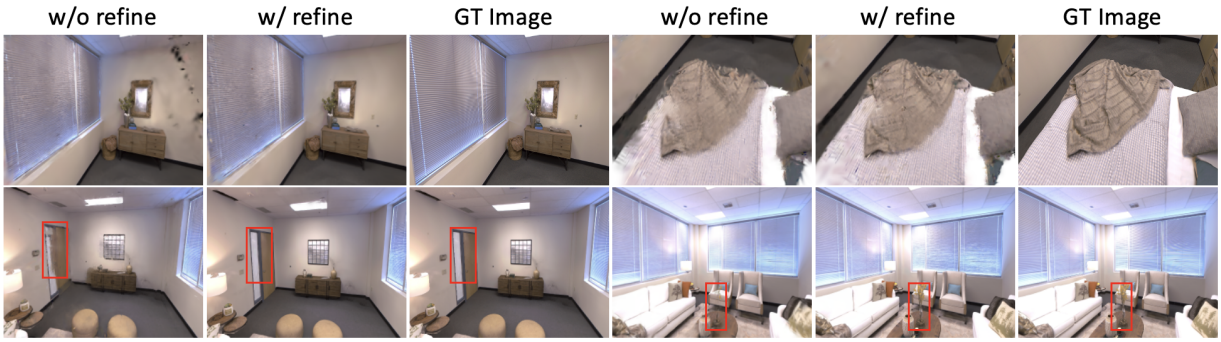


FIGURE 10. **Qualitative comparison of results with and without the depth refinement filter (as described in Section III.B).**

Lastly, the performance of 3DGS modeling is highly influenced by localization errors. To mitigate this issue, we integrated DROID-SLAM [36], a state-of-the-art system, and achieved relatively stable localization accuracy in our experimental datasets, resulting in satisfactory modeling outcomes. However, in challenging scenarios such as textureless scenes or dynamic environments, localization errors are inevitable and can significantly degrade the quality of the reconstructed 3D model. To address this problem, we plan to integrate global dense bundle adjustment (DBA) [36] into our framework in future work. This approach refines the poses of all keyframes while simultaneously adjusting Gaussian points based on the updated poses to ensure consistency. Regular execution of global DBA is expected to greatly improve both localization accuracy and overall modeling performance.

VI. CONCLUSION

We have presented a novel online mapping method for high-quality 3DGS modeling. Our approach leverages an online MVS method to generate high-resolution depth maps and accurately initialize Gaussian points in the 3DGS representation. By integrating depth information sequentially from a local time window and applying a comprehensive filtering process, our method effectively removes outliers, leading to highly accurate and temporally consistent depth estimation. This, in turn, enables the generation of precise and detailed 3DGS models.

Furthermore, to ensure online performance, we introduced an efficient mapping framework that includes a parallelized backend module dedicated to optimizing the 3DGS model while the frontend module handles camera tracking and depth estimation. This parallel processing allows the 3DGS model to be continuously refined and updated with each incoming keyframe, significantly enhancing both rendering accuracy and reconstruction quality. Unlike previous methods that rely on sparse or noisy depth estimates, our framework generates dense, high-fidelity depth maps, resulting in superior Gaussian initialization and more detailed reconstructions.

Through extensive experiments on both indoor and outdoor datasets, we demonstrated that our method outperforms state-of-the-art dense SLAM techniques. In particular, our method performs exceptionally well in challenging outdoor environments where existing approaches face difficulties due to uncertainties in camera tracking and depth estimation. The integration of robust depth filtering and online optimization allows our method to achieve exceptional rendering quality, closely resembling real-world images.

As future work, our technology can be applied to various domains, including digital twins [44] and autonomous robot perception [45] Digital twin systems require continuous 3D model updates to accurately reflect dynamic environmental changes. However, conventional platforms [46] often struggle to meet real-time requirements due to the lengthy process involved in generating and updating 3D models. Our approach overcomes these limitations by enabling the online generation and updating of accurate 3D models directly from RGB camera streams. For instance, in a logistics warehouse setting [47], multiple mobile robots equipped with RGB cameras can continuously capture visual data, which is immediately converted into precise digital twins. These digital models support real-time tasks such as inventory tracking, stock localization, and anomaly detection. Thanks to its accuracy and responsiveness, our method can be effectively extended to a wide range of real-time monitoring and control applications beyond the logistics domain. To support broader adoption in such digital twin applications, however, the system must be further optimized and lightweighted for efficient onboard execution.

REFERENCES

[1] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. 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*. Springer, 2016, pp. 501–518.

[2] X. Gu, Z. Fan, S. Zhu, Z. Dai, F. Tan, and P. Tan, "Cascade cost volume for high-resolution multi-view stereo and stereo matching," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2495–2504.

[3] C. Cao, X. Ren, and Y. Fu, "Mvsformer: Multi-view stereo by learning robust image features and temperature-based depth," *arXiv preprint arXiv:2208.02541*, 2022.

[4] 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.

[5] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo,"

in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 14 124–14 133.

[6] T. Liu, X. Ye, M. Shi, Z. Huang, Z. Pan, Z. Peng, and Z. Cao, "Geometry-aware reconstruction and fusion-refined rendering for generalizable neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 7654–7663.

[7] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[8] S. Zheng, B. Zhou, R. Shao, B. Liu, S. Zhang, L. Nie, and Y. 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*, 2024, pp. 19 680–19 690.

[9] T. Liu, G. Wang, S. Hu, L. Shen, X. Ye, Y. Zang, Z. Cao, W. Li, and Z. Liu, "Fast generalizable gaussian splatting reconstruction from multi-view stereo," *arXiv preprint arXiv:2405.12218*, 2024.

[10] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2320–2327.

[11] M. Pizzoli, C. Forster, and D. Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 2609–2616.

[12] L. Koestler, N. Yang, N. Zeller, and D. Cremers, "Tandem: Tracking and dense mapping in real-time using deep multi-view stereo," in *Conference on Robot Learning*. PMLR, 2022, pp. 34–45.

[13] A. Eftekhar, A. Sax, J. Malik, and A. Zamir, "Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 786–10 796.

[14] C.-M. Chung, Y.-C. Tseng, Y.-C. Hsu, X.-Q. Shi, Y.-H. Hua, J.-F. Yeh, W.-C. Chen, Y.-T. Chen, and W. H. Hsu, "Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9400–9406.

[15] H. Matsuki, E. Sucar, T. Laidow, K. Wada, R. Scona, and A. J. Davison, "imode: Real-time incremental monocular dense mapping using neural field," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 4171–4177.

[16] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 3437–3444.

[17] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, "Go-slam: Global optimization for consistent 3d instant reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3727–3737.

[18] Z. Zhu, S. Peng, V. Larsson, Z. Cui, M. R. Oswald, A. Geiger, and M. Pollefeys, "Nicer-slam: Neural implicit scene encoding for rgb slam," in *2024 International Conference on 3D Vision (3DV)*. IEEE, 2024, pp. 42–52.

[19] G. Zhang, E. Sandström, Y. Zhang, M. Patel, L. Van Gool, and M. R. Oswald, "Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam," *arXiv preprint arXiv:2403.19549*, 2024.

[20] C. Peng, C. Xu, Y. Wang, M. Ding, H. Yang, M. Tomizuka, K. Keutzer, M. Pavone, and W. Zhan, "Q-slam: Quadric representations for monocular slam," *arXiv preprint arXiv:2403.08125*, 2024.

[21] H. Huang, L. Li, H. Cheng, and S.-K. 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*, 2024, pp. 21 584–21 593.

[22] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison, "Gaussian splatting slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 039–18 048.

[23] T. Lan, Q. Lin, and H. Wang, "Monocular gaussian slam with language extended loop closure," *arXiv preprint arXiv:2405.13748*, 2024.

[24] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 12 786–12 796.

[25] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. 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*, 2024, pp. 21 357–21 366.

[26] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[27] A. Hamdi, L. Melas-Kyriazi, J. Mai, G. Qian, R. Liu, C. Vondrick, B. Ghanem, and A. Vedaldi, "Ges: Generalized exponential splatting for efficient radiance field rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 812–19 822.

[28] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 573–580.

[29] 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.

[30] K. T. Giang, S. Song, and S. Jo, "Curvature-guided dynamic scale networks for multi-view stereo," *arXiv preprint arXiv:2112.05999*, 2021.

[31] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[32] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE transactions on robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[33] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[34] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

[35] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 767–783.

[36] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.

[37] N. Burgdorfer and P. Mordohai, "V-fuse: Volumetric depth map fusion with long-range constraints," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3449–3458.

[38] S. Song, D. Kim, and S. Choi, "View path planning via online multiview stereo for 3-d modeling of large-scale structures," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 372–390, 2021.

[39] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[40] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5828–5839.

[41] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[42] F. Wang, S. Galliani, C. Vogel, P. Speciale, and M. Pollefeys, "Patchmatchnet: Learned multi-view patchmatch stereo," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14 194–14 203.

[43] F. Wang, S. Galliani, C. Vogel, and M. Pollefeys, "Itermvs: Iterative probability estimation for efficient multi-view stereo," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8606–8615.

[44] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on industrial informatics*, vol. 15, no. 4, pp. 2405–2415, 2018.

[45] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5664–5673.

[46] A. Barbasiewicz, T. Widerski, and K. Daliga, "The analysis of the accuracy of spatial models using photogrammetric software: Agisoft photoscan and pix4d," in *E3S Web of Conferences*, vol. 26. EDP Sciences, 2018, p. 00012.

[47] P. Stączek, J. Pizoń, W. Danilczuk, and A. Gola, "A digital twin approach for the improvement of an autonomous mobile robots (amr's) operating environment—a case study," *Sensors*, vol. 21, no. 23, p. 7830, 2021.

BYEONGGWON LEE received the B.S. degree in software engineering from Hanyang University, South Korea, in 2023. He is currently pursuing the M.S. degree in computer science and artificial intelligence with Dongguk University, Seoul, South Korea. His current research interests include 3D reconstruction, computer vision, and SLAM.

JUNKYU PARK is currently pursuing the B.S. degree in computer science at Dongguk University, Seoul, South Korea, while working as an undergraduate researcher at the AI & Robotics Lab. His research interests include computer vision, 3D reconstruction, and visual SLAM.

KHANG TRUONG GIANG received the B.S. degree from Hanoi University of Science and Technology (HUST), Vietnam, in 2018, and the M.S. and Ph.D. degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST), in 2021 and 2024, respectively. His research interests include 3D reconstruction, feature matching, visual localization, and multi-view depth estimation.

SUNGHO JO (Senior Member, IEEE) received the B.S. degree in mechanical and aerospace engineering from the School of Mechanical and Aerospace Engineering, Seoul National University, Seoul, South Korea, in 1999, and the S.M. degree in mechanical engineering and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1999 and 2006, respectively. While pursuing the Ph.D. degree, he was associated with the Computer Science and Artificial Intelligence Laboratory (CSAIL) and the Laboratory for Information Decision and Systems (LIDS) and Harvard-MIT HST NeuroEngineering Collaborative. Before joining as a Faculty Member of KAIST, he was a Postdoctoral Researcher with the MIT Media Laboratory. Since December 2007, he has been with the School of Computing, KAIST, where he is currently a Full Professor. He is also associated with the KAIST Institute for AI and the KAIST Institute for Robotics. He is also an Adjunct Professor with the KAIST Robotics Program. His research interests include robotic intelligence, augmented intelligence, and neuro-hybrid intelligence.

SOOHWAN SONG received the B.S. degree in information and communication engineering from Dongguk University, Seoul, South Korea, in 2013, and the M.S. and Ph.D. degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2015 and 2020, respectively. He is currently an Assistant Professor with the College of AI Convergence, Dongguk University. Prior to joining Dongguk University, he was a Senior Researcher with the Field Robotics Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon. His research interests include robotics, motion planning, multi-robot systems, and computer vision.